INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

ROBERT GORDON UNIVERSITY ABERDEEN

Department of Computing

Name: A.M Sandushke De Alwis

IIT number: 20200244

RGU number : 2117852

Module: CM2602 Artificial Intelligence

Module Leader: Mr. Nipuna Senanayake

Handout Date: 3rd Week of October

Submission Date: 31st of December 2022

Acknowledge

I would like to take this opportunity to thank all the IIT staff for giving us the opportunity to follow the degree artificial intelligence & data science and also l specially thank my lecturer and tutor Mr.Nipuna Senanayake for giving me the knowledge to do this report. Rather than the academic knowledge I got a lot of information while gathering information. So, at last, I would like to take the opportunity to thank for all the feedbacks that you provided for me and it really helped me to correct all our mistakes. Thank you

Question 01

a)

1. Functionality of Naïve Auction algorithm:

The objective of Naïve Auction algorithm is to provide set of tasks to agents in a way that it fulfils a set of requirements of optimization criteria. This method is followed to solve multi agent assignment issues

As an example, the below code shows a pseudocode that is used for naive auction algorithm when comparing winning bids out of a list of bids

Pseudocode for Naïve auction Algorithm:

```
function naiveAuction(bids: List[int]):
    bidsWin = []
        for i in bid:
            if (i > currentWiningBid):
                    currentWiningBid = i
                    bidsWin.append(i)
return bidsWin
```

2. Issues with Naive Auction Algorithm:

Naïve auction is a type of auction algorithm which is commonly used to allocate resources to a distributed system. However, this algorithm has issues where few of them are listed below.

- Naïve auction can be slow and inefficient when working with large number of complex allocation instances. This is because this algorithm uses a bidding process where it reduces the overall efficiency of the process

- Another issue with this algorithm is that, it has a tendency for cycling which could happen when an algorithm gets stuck in a loop. It will keep switching between one or more solutions continuously without giving an ideal solution. This could lead the algorithm to run unnecessarily for a long time.

3. Ways to rectify these issues:

- One way to solve this issue is to use a modified version of the naïve auction algorithm that would add techniques like stimulated annealing or genetic algorithms that would help to overcome the solution and find better solutions.

- Another solution to solve this issue is to solve the subgoals with more effective algorithms. Hungarian algorithm, Successive shortest path algorithm are few of them.

Algorithm for the flowchart:

1. Start
2. Set all the values of the needed items to zero
3. Set assignment to all unassigned items
4. For each item, find the agent who values an item the most for each one
    a) If the item is not already assigned, assign it to the agent that values the most
    b) If it is not assigned, check whether the current agents' value is more than the agent that values the most
    c) Reassign the item to a new agent

5.Repeat step 2-4 until all items are assigned

6. End

b)

1.

2. Netlogo model for virus behavior is uploaded in the folder

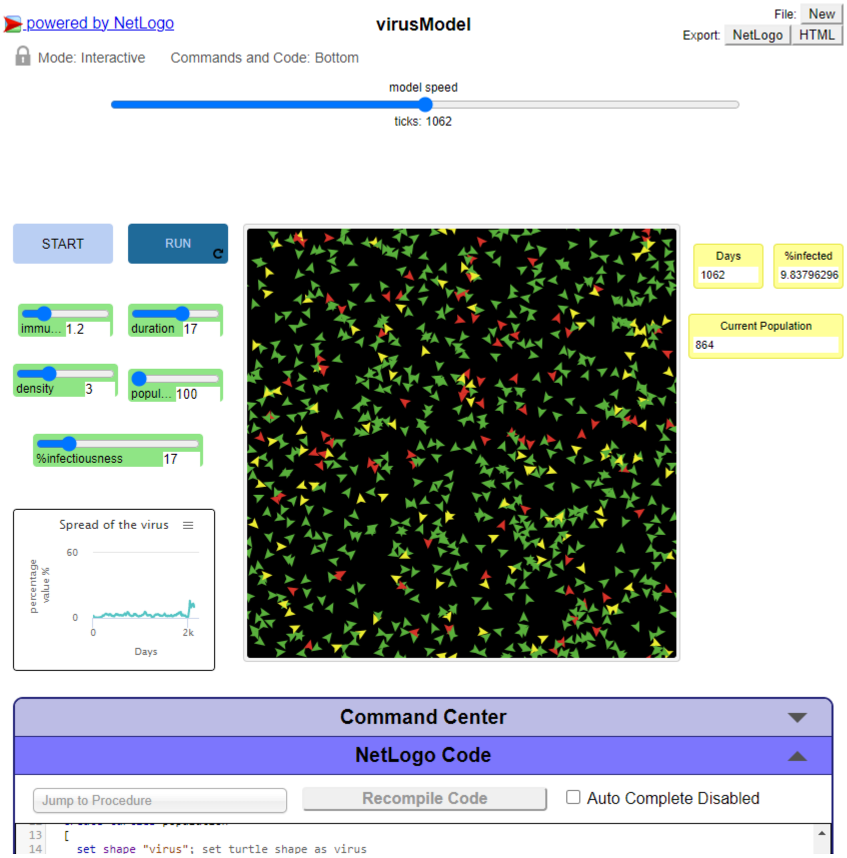
3.

Code snippet for virusModel:

```
;;create three 3 global variables as infected, recover and current population
globals [
  %infected
  survive
  current-population
]

;; work needed to be done by the start button
to start
  clear-all; reset model everytime it runs from the beginning
  reset-ticks; reset turtles everytime it runs from the beginning
  create-turtles population
  [
    set shape "virus"; set turtle shape as virus
    set color green; set healthy turtle color as green
    setxy random-xcor random-ycor; set random x and y cordinates for the turtle
  ]

  ask turtle 3 [set color red]; set three turtles as infected viruses
  repeat 10
  [
    ask turtle random 100 [set color yellow];; out of all turtles, set 100 immune turtles randomly
  ]
  set %infected (count turtles with [color = red] / count turtles) * 100
  set population 100
end

;; work needed to be done by the go button
to go
  tick
  ask turtles
  [rt random 100 lt random 100 forward 0.2];move turtles with a speed of 0.5s
  ask turtles with [color = red]
    [
      ask turtles-here with [color = green]
      [if (random 100 * immunity) < %infectiousness
        [set color red]
      ]
  ]
  set survive random 50
  if survive > duration [;if the recovery rate is > 50, make the infected virus die
    ask turtles with [color = red]
    [
      repeat 2
      [
        ask one-of turtles [die];make the infected virus die automatically
      ]
    ]
    create-turtles 20
      [
        set shape "virus"
        set color green
        setxy random-xcor random-ycor
      ]
    create-turtles 2;;create 2 immuned turtles
      [
        set shape "virus"
        set color yellow
        setxy random-xcor random-ycor;set random x and y cordinates for the turtle
      ]
  ]

  ;; if the no. of days reaches more than 1000, create a new generation
  if ticks > 1000
  [
    create-turtles population
    [
      set shape "virus"
      set color green
      setxy random-xcor random-ycor
    ]
    create-turtles 8
    [
      set shape "virus"
      set color red
      setxy random-xcor random-ycor
    ]
    create-turtles 15
    [
      set shape "virus"
      set color yellow
      setxy random-xcor random-ycor
    ]
  ]
  ;;display all the stats
```

```
    set %infected (count turtles with [color = red] / count turtles) * 100
    set current-population count turtles with [color = green] + count turtles with [color = red]
    + count turtles with [color = yellow]
  end
```

Screenshots of the code and virus model:



ticks: 1062

immu... 1.2

duration 17

density 3

popul... 100

%infectiousness 17

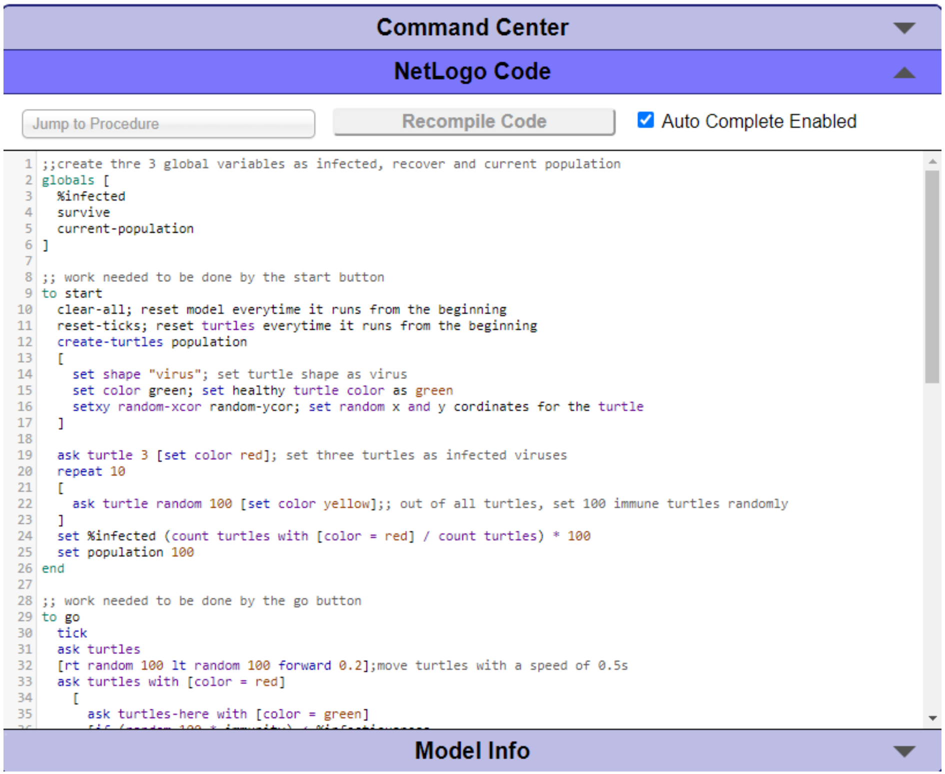
Days 1062

%infected 9.83796296

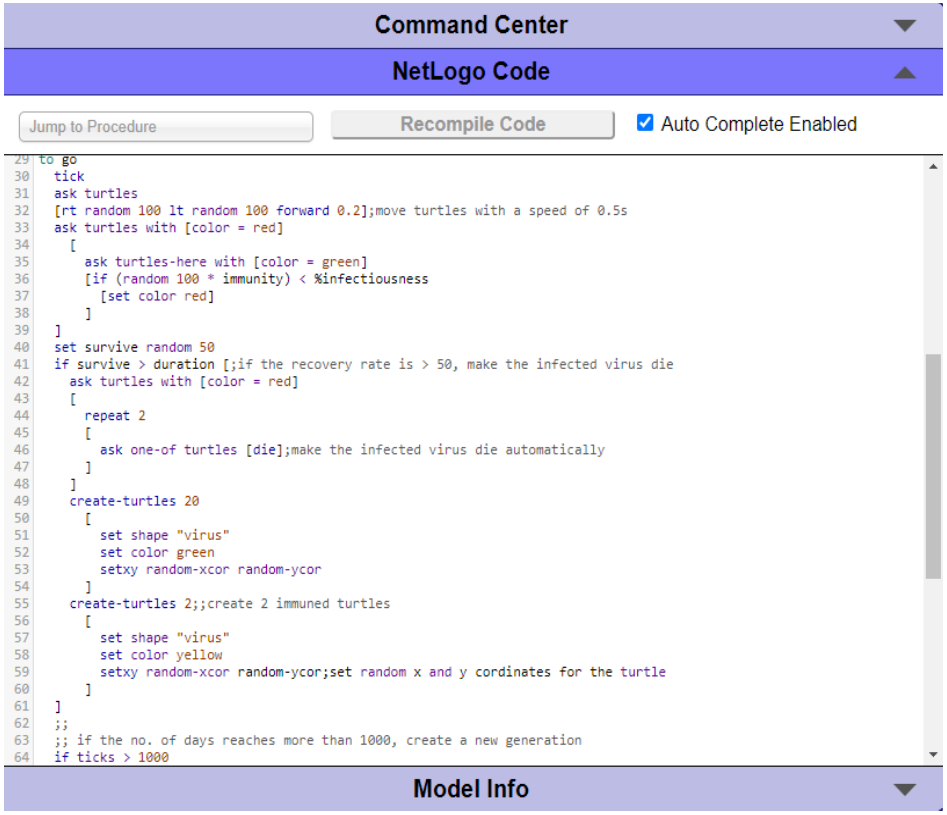
Current Population 864

□ Auto Complete Disabled

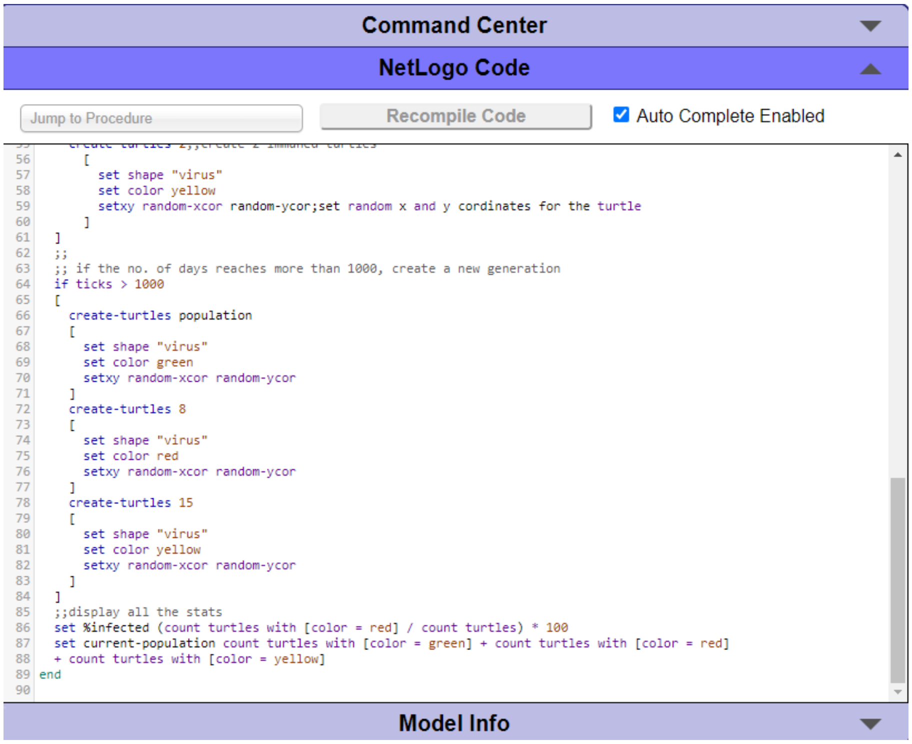
```
[
  set shape "virus"; set turtle shape as virus
```



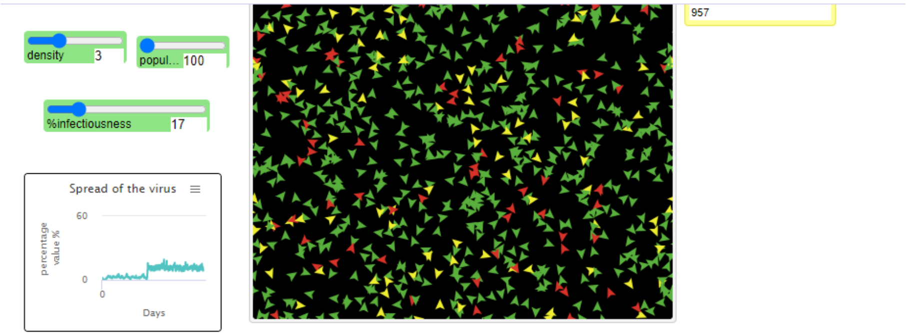
```
;;create thre 3 global variables as infected, recover and current population
globals [
  %infected
  survive
  current-population
]

;; work needed to be done by the start button
to start
  clear-all; reset model everytime it runs from the beginning
  reset-ticks; reset turtles everytime it runs from the beginning
  create-turtles population
  [
    set shape "virus"; set turtle shape as virus
    set color green; set healthy turtle color as green
    setxy random-xcor random-ycor; set random x and y cordinates for the turtle
  ]

  ask turtle 3 [set color red]; set three turtles as infected viruses
  repeat 10
  [
    ask turtle random 100 [set color yellow];; out of all turtles, set 100 immune turtles randomly
  ]
  set %infected (count turtles with [color = red] / count turtles) * 100
  set population 100
end

;; work needed to be done by the go button
to go
  tick
  ask turtles
  [rt random 100 lt random 100 forward 0.2];move turtles with a speed of 0.5s
  ask turtles with [color = red]
    [
      ask turtles-here with [color = green]
```



```
to go
  tick
  ask turtles
  [rt random 100 lt random 100 forward 0.2];move turtles with a speed of 0.5s
  ask turtles with [color = red]
    [
      ask turtles-here with [color = green]
      [if (random 100 * immunity) < %infectiousness
        [set color red]
      ]
  ]
  set survive random 50
  if survive > duration [;if the recovery rate is > 50, make the infected virus die
    ask turtles with [color = red]
    [
      repeat 2
      [
        ask one-of turtles [die];make the infected virus die automatically
      ]
    ]
    create-turtles 20
      [
        set shape "virus"
        set color green
        setxy random-xcor random-ycor
      ]
    create-turtles 2;;create 2 immuned turtles
      [
        set shape "virus"
        set color yellow
        setxy random-xcor random-ycor;set random x and y cordinates for the turtle
      ]
  ]
  ;;
  ;; if the no. of days reaches more than 1000, create a new generation
  if ticks > 1000
```



```
    create-turtles 2;;create 2 immuned turtles
      [
        set shape "virus"
        set color yellow
        setxy random-xcor random-ycor;set random x and y cordinates for the turtle
      ]
  ]
  ;;
  ;; if the no. of days reaches more than 1000, create a new generation
  if ticks > 1000
  [
    create-turtles population
    [
      set shape "virus"
      set color green
      setxy random-xcor random-ycor
    ]
    create-turtles 8
    [
      set shape "virus"
      set color red
      setxy random-xcor random-ycor
    ]
    create-turtles 15
    [
      set shape "virus"
      set color yellow
      setxy random-xcor random-ycor
    ]
  ]
  ;;display all the stats
  set %infected (count turtles with [color = red] / count turtles) * 100
  set current-population count turtles with [color = green] + count turtles with [color = red]
  + count turtles with [color = yellow]
end
```

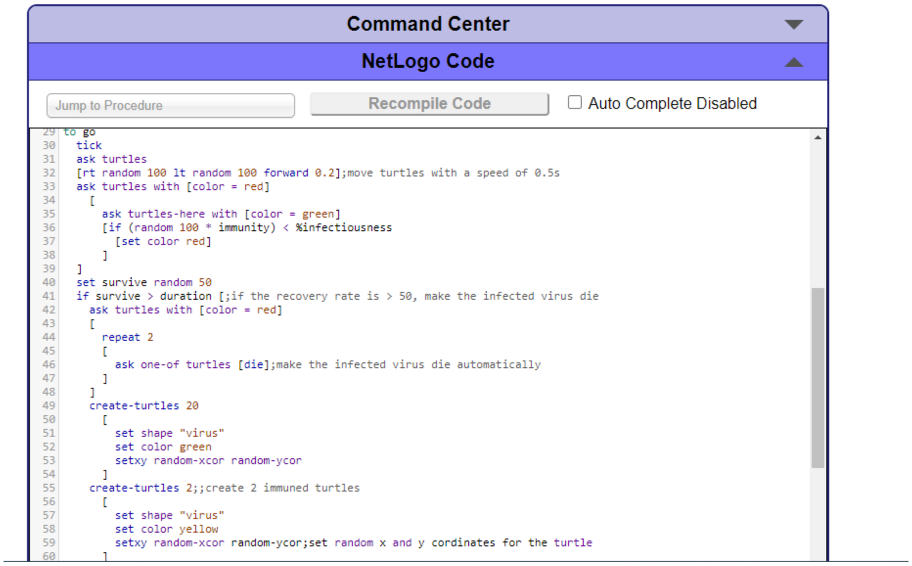


density 3

popul... 100

%infectiousness 17

957



```
to go
  tick
  ask turtles
  [rt random 100 lt random 100 forward 0.2];move turtles with a speed of 0.5s
  ask turtles with [color = red]
    [
      ask turtles-here with [color = green]
      [if (random 100 * immunity) < %infectiousness
        [set color red]
      ]
  ]
  set survive random 50
  if survive > duration [;if the recovery rate is > 50, make the infected virus die
    ask turtles with [color = red]
    [
      repeat 2
      [
        ask one-of turtles [die];make the infected virus die automatically
      ]
    ]
    create-turtles 20
      [
        set shape "virus"
        set color green
        setxy random-xcor random-ycor
      ]
    create-turtles 2;;create 2 immuned turtles
      [
        set shape "virus"
        set color yellow
        setxy random-xcor random-ycor;set random x and y cordinates for the turtle
      ]
```

Explanation of the virus model and the code snippet and functionalities:

Virus model is a type of a simulation model that shows the spread of a virus and the survival stability of human population.

According to the simulation model that was created, 6 parameters were identified whereas 5 of the functionalities are briefly explained below.

1. immunity - shows the immunity level of the population
2. density – shows how fast the population moves
3. population – shows the no. of population in the simulation model
4. infectiousness – shows the speed that a virus spreads to a healthy human being
5. duration – shows the time period taken

The three global variables "infected," "survive," and "current population" were used to develop the virus model. A healthy turtle is shaped like a virus and has the color green, whereas an infected turtle has the color red, and an immune turtle has the color yellow. In the simulation model, random turtles are placed in random x, y coordinates. Out of 100 turtles, 3 at random were created as infected viruses at first, and a loop was then used to repeat 10 times in order to identify and set immune turtles (yellow). If an infected turtle gets contact with a healthy turtle, the healthy turtle will get infect so that its color will turn into red. Also if the immunity level of a turtle is less than the infected level, that particular turtle will become an infected turtle (red). Although, if an infected turtle takes a long time period to survive than it was expected recover, then that turtle is set in a way to die according to the nature.

However, once the number of days exceeds more than 1000, the model will create a new generation where it will have both the previous healthy population of the old generation and the newly created human population. The model is generated in a way that when a new generation is formed, all the turtles should be as how it was assigned in the beginning. The graph in the model will keep plotting the change of the virus spread and also this model will keep monitoring and displaying all the changes that will happen throughout the whole simulation process.

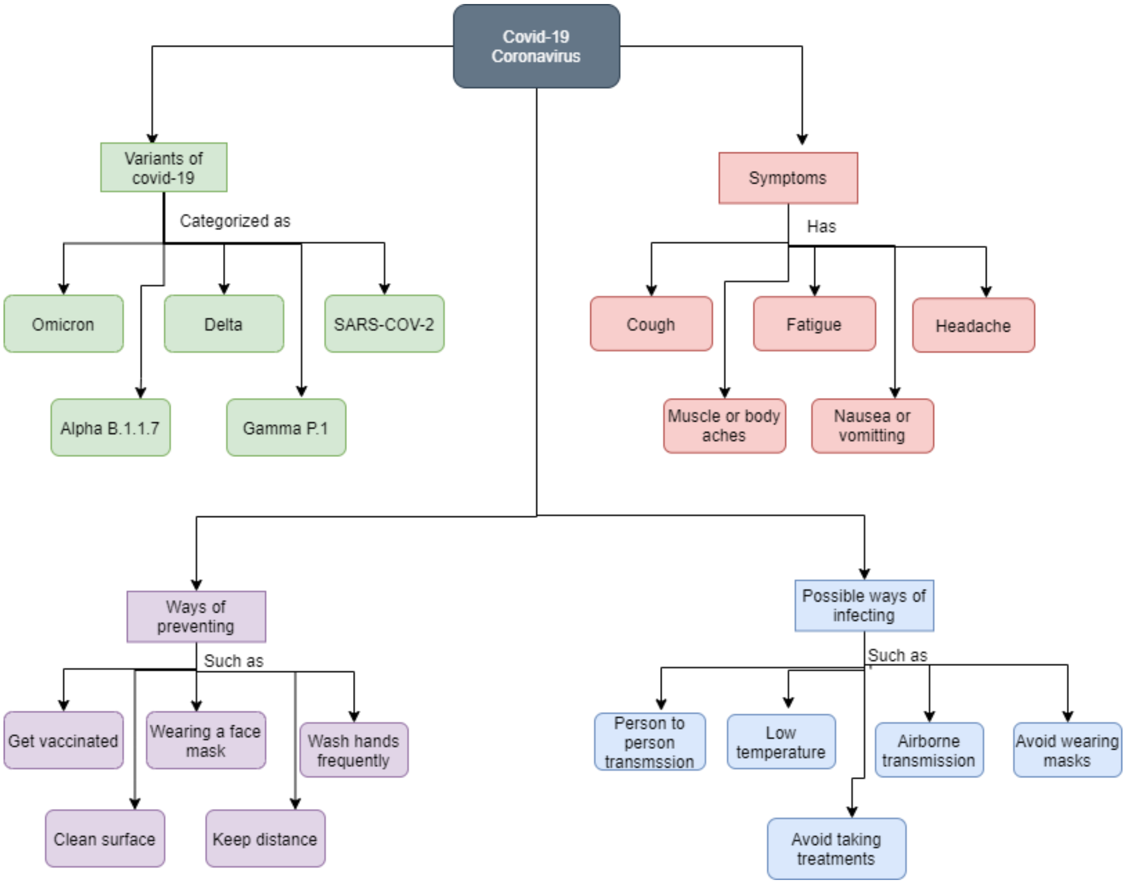
Question 02

1. The scope of the knowledge base depicts the aspects like variants of the disease, symptoms, prevention and control measures, treatments and impacts of COVID-19 (Coronavirus)

   Competency Questions for Covid-19:

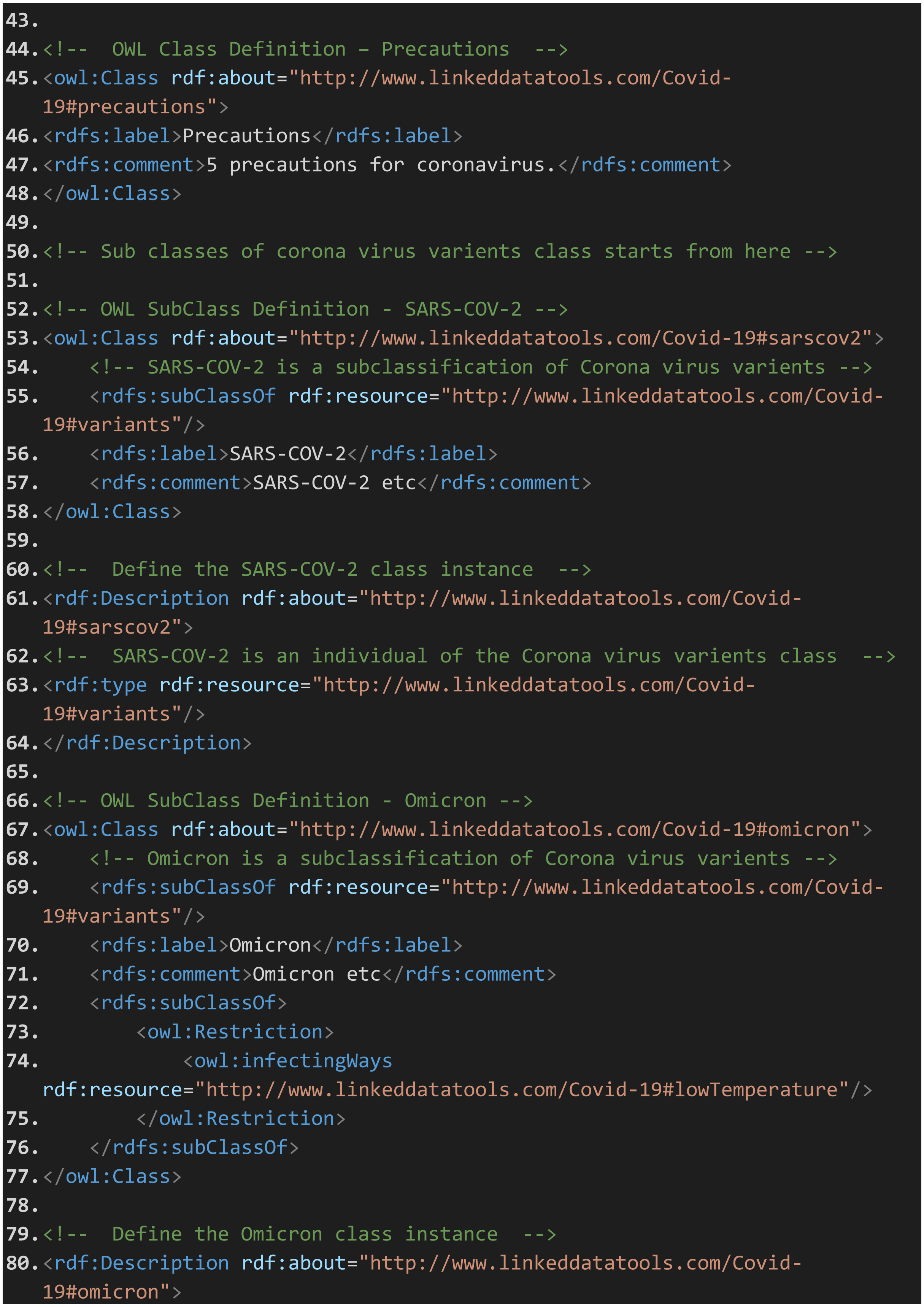
i. What are the ways that people can get infect of covid-19?
ii. What are the symptoms of covid-19?j
iii. What are the current treatment options for covid-19?
iv. How can covid-19 be prevented among people ?
v. What are the variants of covid-19?

2. Concept graph:

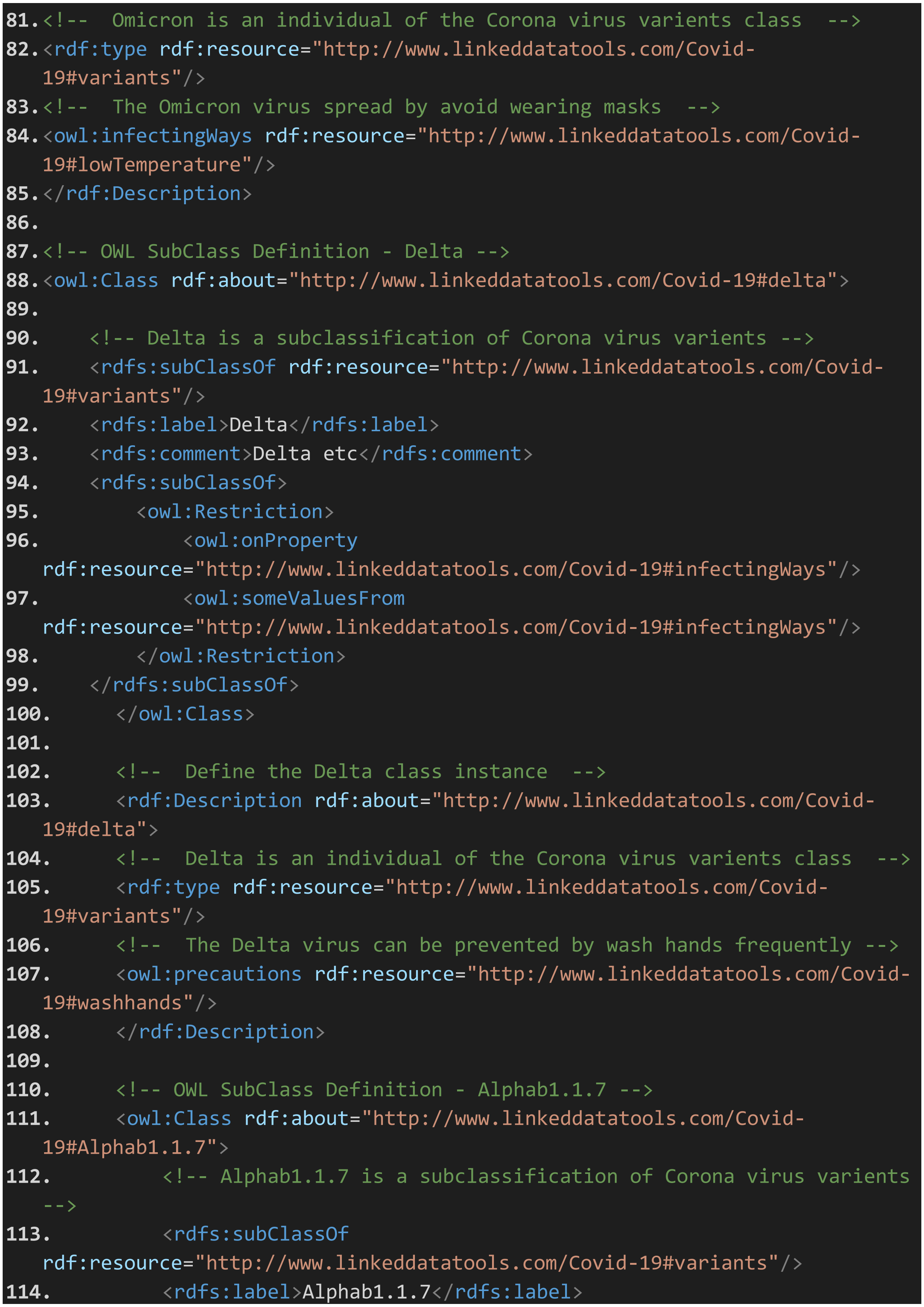


3. XML file for coronavirus is uploaded in the folder.

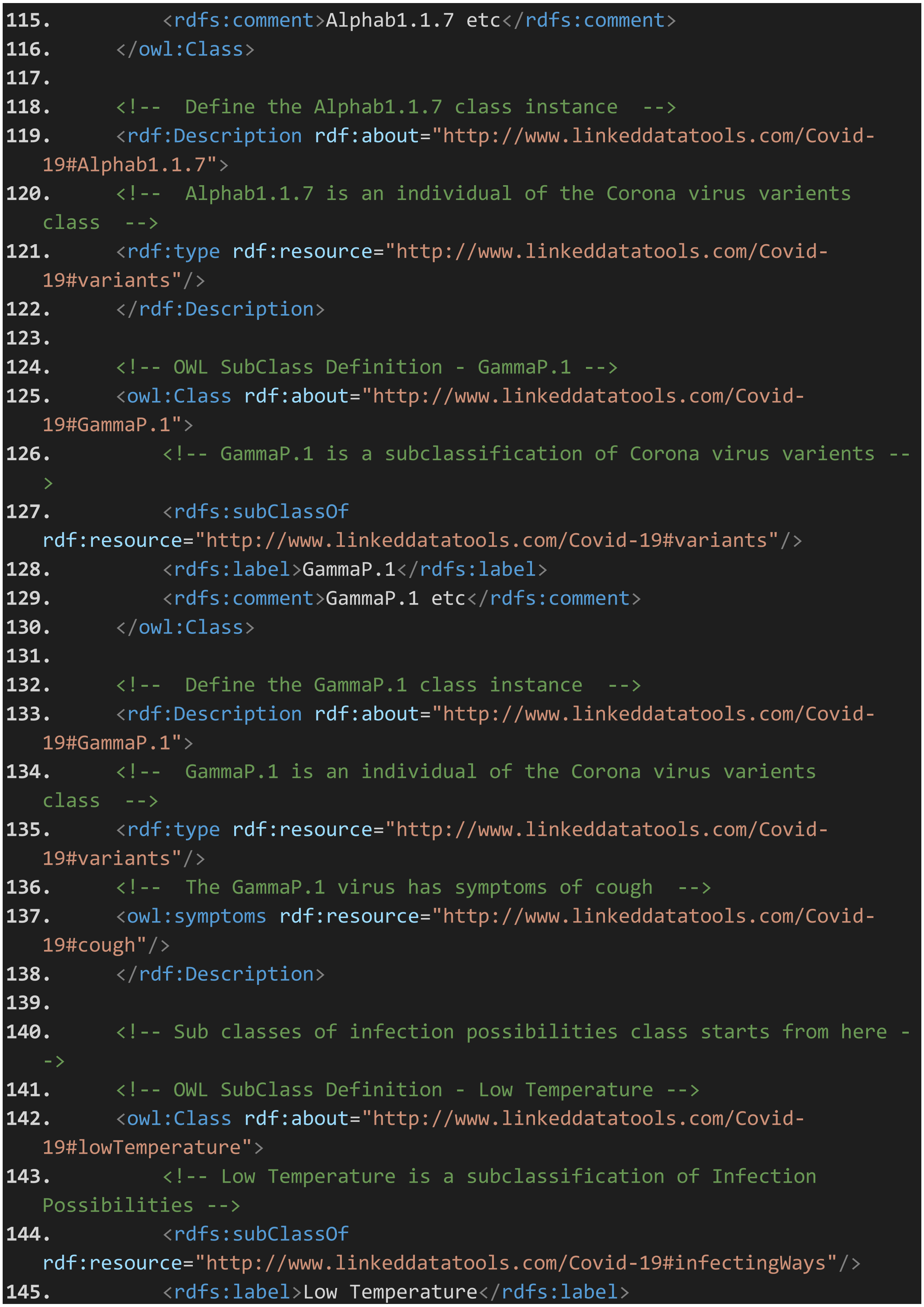
```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:dc="http://purl.org/dc/elements/1.1/">

<!--  OWL Class Definition - variants of coronavirus  -->
<!--  OWL Header  -->
<owl:Ontology rdf:about="http://www.linkeddatatools.com/Covid-19">
<dc:title>  Ontology for Coronavirus(Covid-19)</dc:title>
<dc:description>Building an ontology for Coronavirus using RDF</dc:description>
</owl:Ontology>

<!--  Define the infecting way By property  -->
<owl:ObjectProperty rdf:about="http://www.linkeddatatools.com/Covid-19#infectingWays"/>

<!--  Define the precautions property  -->
<owl:ObjectProperty rdf:about="http://www.linkeddatatools.com/Covid-19#precautions"/>

<!--  Define the symptoms property  -->
<owl:ObjectProperty rdf:about="http://www.linkeddatatools.com/Covid-19#symptoms"/>

<!--  OWL Class Definition - variants of coronavirus  -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#variants">
<rdfs:label>Varients of Coronavirus </rdfs:label>
<rdfs:comment>5 variant types of coronavirus</rdfs:comment>
</owl:Class>

<!--  OWL Class Definition – Infection Ways  -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#infectingWays">
<rdfs:label>Infecting ways</rdfs:label>
<rdfs:comment>5 infecting ways of coronavirus</rdfs:comment>
</owl:Class>

<!--  OWL Class Definition – Symptoms  -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#symptoms">
<rdfs:label>Symptoms</rdfs:label>
<rdfs:comment>5 symptoms of coronavirus </rdfs:comment>
</owl:Class>
```



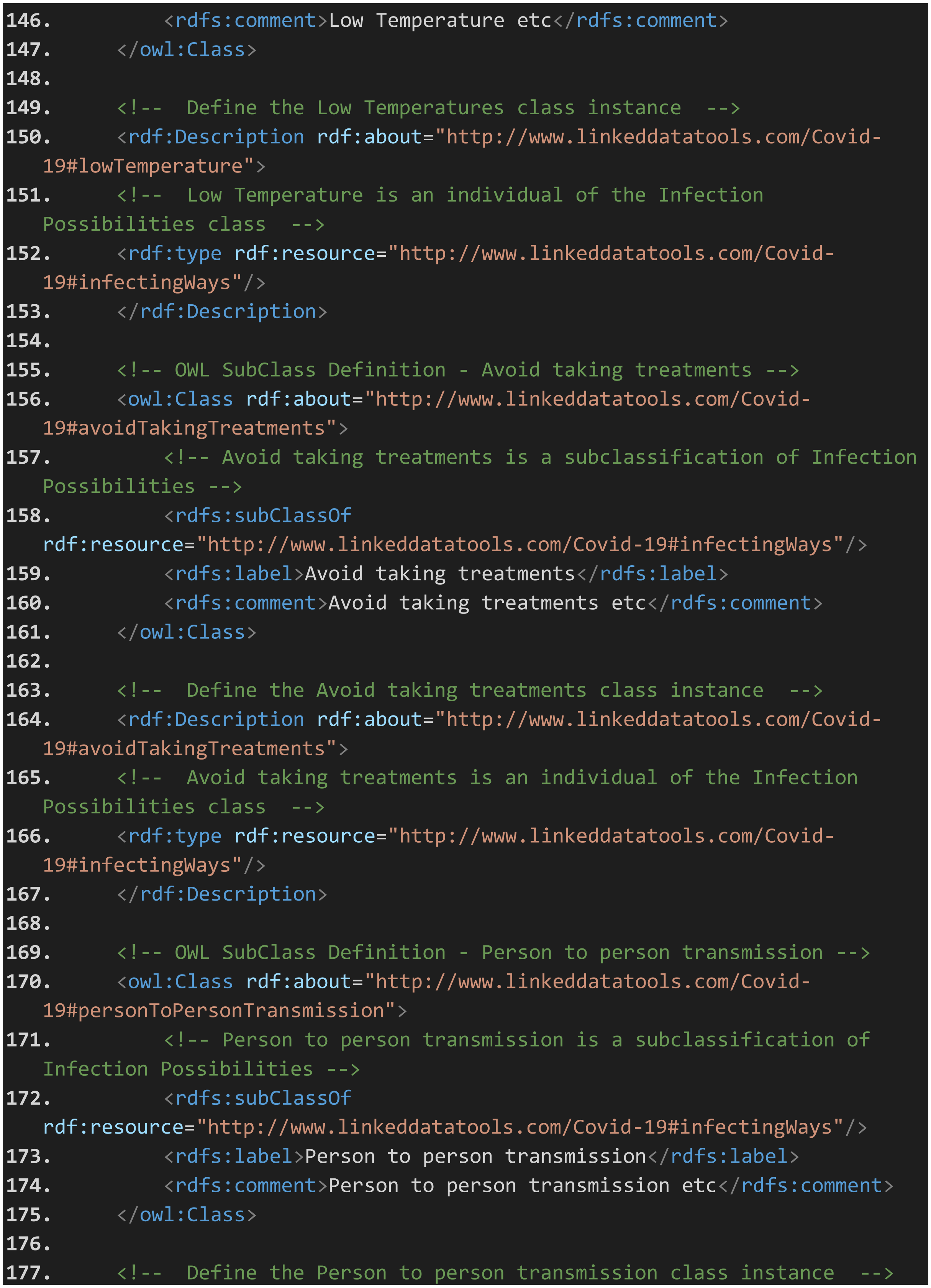
```xml

<!--  OWL Class Definition – Precautions  -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#precautions">
<rdfs:label>Precautions</rdfs:label>
<rdfs:comment>5 precautions for coronavirus.</rdfs:comment>
</owl:Class>

<!-- Sub classes of corona virus varients class starts from here -->

<!-- OWL SubClass Definition - SARS-COV-2 -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#sarscov2">
    <!-- SARS-COV-2 is a subclassification of Corona virus varients -->
    <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
    <rdfs:label>SARS-COV-2</rdfs:label>
    <rdfs:comment>SARS-COV-2 etc</rdfs:comment>
</owl:Class>

<!--  Define the SARS-COV-2 class instance  -->
<rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#sarscov2">
<!--  SARS-COV-2 is an individual of the Corona virus varients class  -->
<rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
</rdf:Description>

<!-- OWL SubClass Definition - Omicron -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#omicron">
    <!-- Omicron is a subclassification of Corona virus varients -->
    <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
    <rdfs:label>Omicron</rdfs:label>
    <rdfs:comment>Omicron etc</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:infectingWays rdf:resource="http://www.linkeddatatools.com/Covid-19#lowTemperature"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<!--  Define the Omicron class instance  -->
<rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#omicron">
```



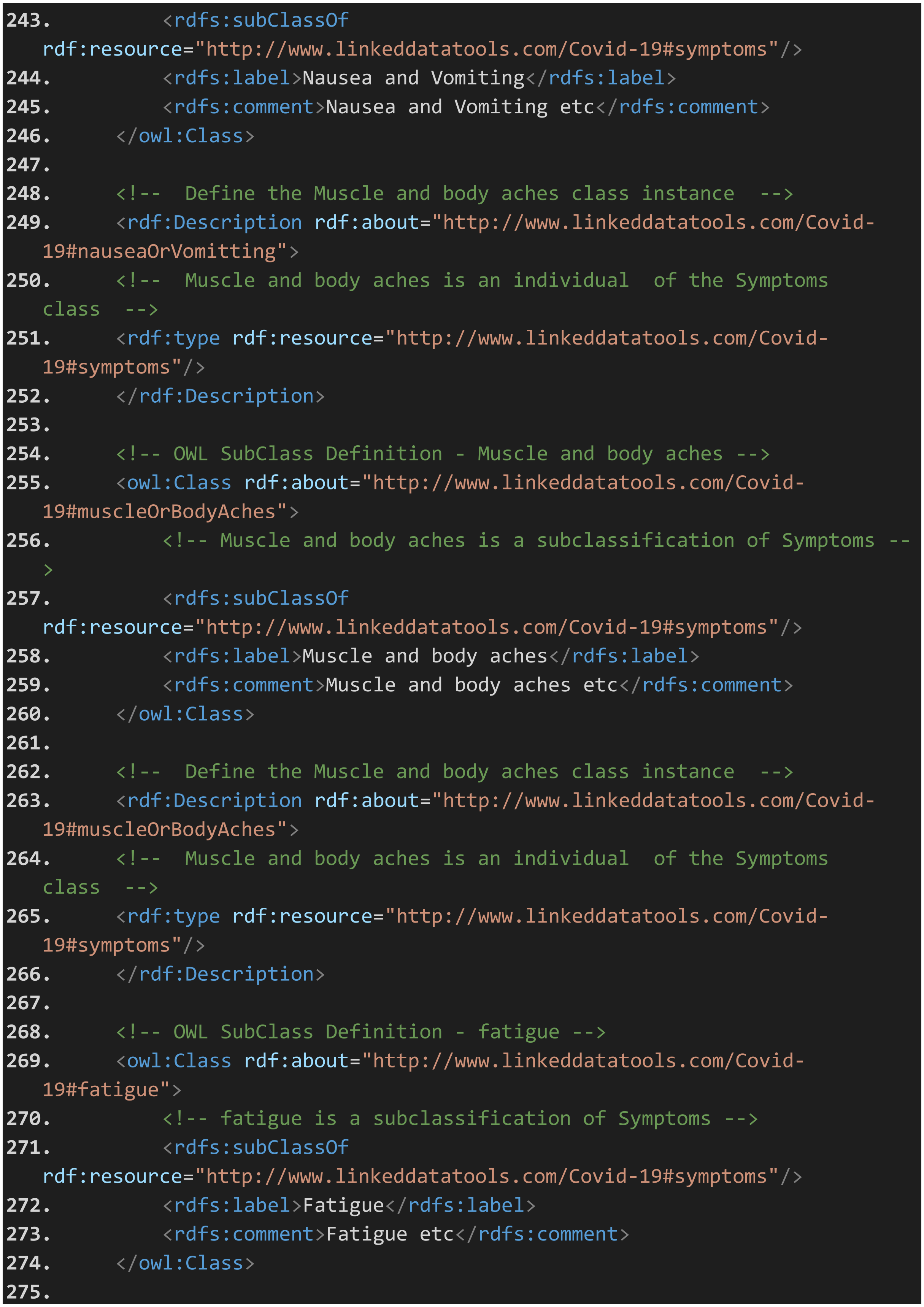
```
<!--  Omicron is an individual of the Corona virus varients class  -->
<rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
<!--  The Omicron virus spread by avoid wearing masks  -->
<owl:infectingWays rdf:resource="http://www.linkeddatatools.com/Covid-19#lowTemperature"/>
</rdf:Description>

<!-- OWL SubClass Definition - Delta -->
<owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#delta">

    <!-- Delta is a subclassification of Corona virus varients -->
    <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
    <rdfs:label>Delta</rdfs:label>
    <rdfs:comment>Delta etc</rdfs:comment>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
            <owl:someValuesFrom rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    </owl:Class>

    <!--  Define the Delta class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#delta">
    <!--  Delta is an individual of the Corona virus varients class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
    <!--  The Delta virus can be prevented by wash hands frequently -->
    <owl:precautions rdf:resource="http://www.linkeddatatools.com/Covid-19#washhands"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Alphab1.1.7 -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#Alphab1.1.7">
        <!-- Alphab1.1.7 is a subclassification of Corona virus varients -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
        <rdfs:label>Alphab1.1.7</rdfs:label>
```



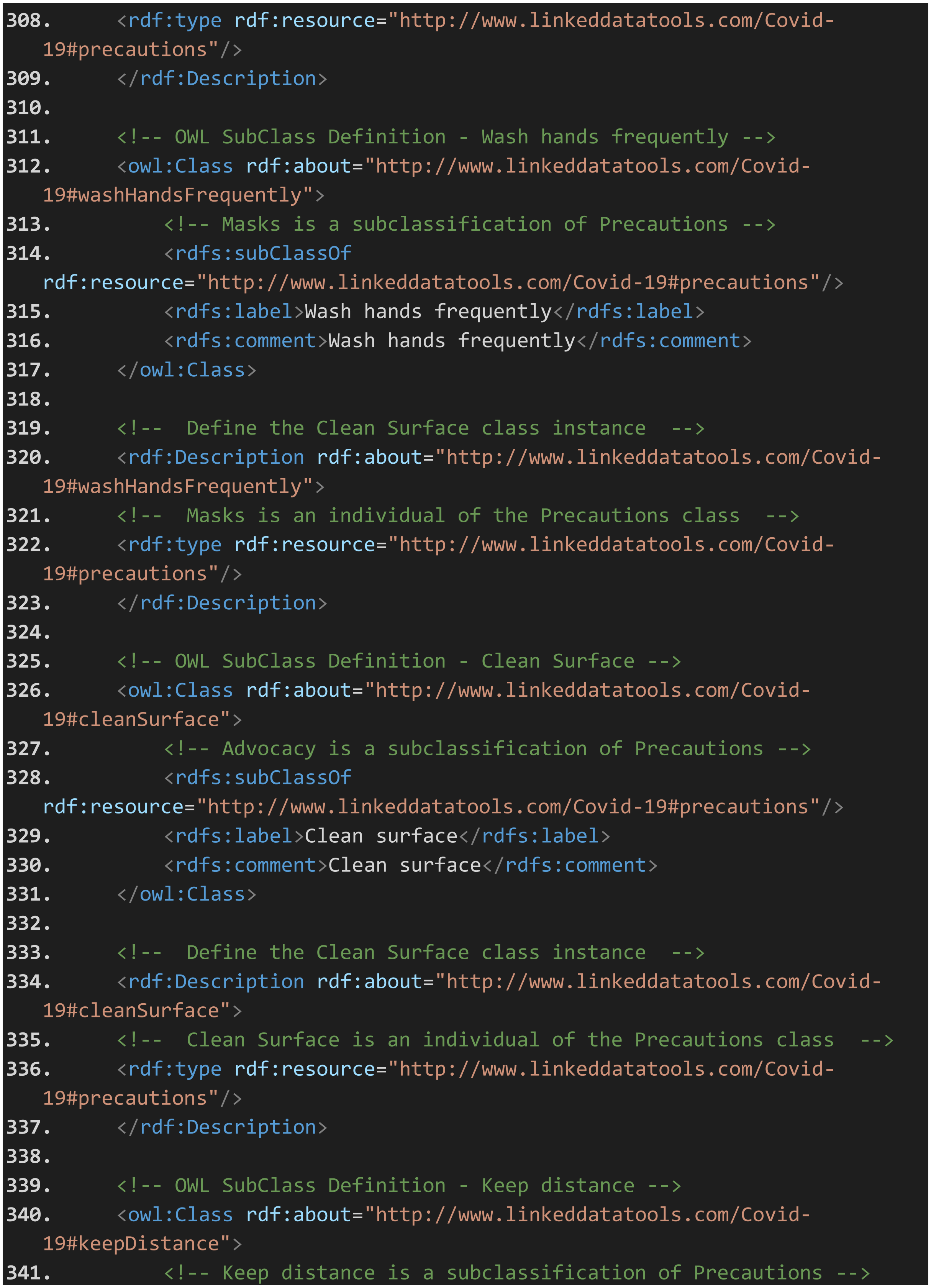
```
        <rdfs:comment>Alphab1.1.7 etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Alphab1.1.7 class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#Alphab1.1.7">
    <!--  Alphab1.1.7 is an individual of the Corona virus varients class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - GammaP.1 -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#GammaP.1">
        <!-- GammaP.1 is a subclassification of Corona virus varients -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
        <rdfs:label>GammaP.1</rdfs:label>
        <rdfs:comment>GammaP.1 etc</rdfs:comment>
    </owl:Class>

    <!--  Define the GammaP.1 class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#GammaP.1">
    <!--  GammaP.1 is an individual of the Corona virus varients class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#variants"/>
    <!--  The GammaP.1 virus has symptoms of cough  -->
    <owl:symptoms rdf:resource="http://www.linkeddatatools.com/Covid-19#cough"/>
    </rdf:Description>

    <!-- Sub classes of infection possibilities class starts from here -->
    <!-- OWL SubClass Definition - Low Temperature -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#lowTemperature">
        <!-- Low Temperature is a subclassification of Infection Possibilities -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
        <rdfs:label>Low Temperature</rdfs:label>
```



```
      <rdfs:comment>Low Temperature etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Low Temperatures class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#lowTemperature">
    <!--  Low Temperature is an individual of the Infection Possibilities class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Avoid taking treatments -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#avoidTakingTreatments">
      <!-- Avoid taking treatments is a subclassification of Infection Possibilities -->
      <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
      <rdfs:label>Avoid taking treatments</rdfs:label>
      <rdfs:comment>Avoid taking treatments etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Avoid taking treatments class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#avoidTakingTreatments">
    <!--  Avoid taking treatments is an individual of the Infection Possibilities class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Person to person transmission -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#personToPersonTransmission">
      <!-- Person to person transmission is a subclassification of Infection Possibilities -->
      <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
      <rdfs:label>Person to person transmission</rdfs:label>
      <rdfs:comment>Person to person transmission etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Person to person transmission class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#personToPersonTransmission">
    <!--  Person to person transmission is an individual of the Infection Possibilities class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Airborn transmission -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#airbornTransmission">
        <!-- avoid tests is a subclassification of Infection Possibilities -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
        <rdfs:label>Airborn transmission</rdfs:label>
        <rdfs:comment>Airborn transmission etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Avoid wearing masks Temperature class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#airbornTransmission">
    <!--  Airborn transmissions is an individual of the Infection Possibilities class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Avoid wearing masks -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#lowtemperature">
        <!-- Avoid wearing masks is a subclassification of Infection Possibilities -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
        <rdfs:label>Avoid wearing masks</rdfs:label>
        <rdfs:comment>Avoid wearing masks etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Avoid wearing masks class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#lowtemperature">
    <!--  Avoid wearing masks is an individual (instance) of the Infection Possibilities class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#infectingWays"/>
    </rdf:Description>

    <!-- Sub classes of symptoms class starts from here -->
    <!-- OWL SubClass Definition - cough -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#cough">
        <!-- cough is a subclassification of Symptoms -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
        <rdfs:label>Cough</rdfs:label>
        <rdfs:comment>Cough etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Cough class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#cough">
    <!--  Cough is an individual of the Symptoms class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - headache -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#headache">
        <!-- headache is a subclassification of Symptoms -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
        <rdfs:label>Headache</rdfs:label>
        <rdfs:comment>Headache etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Headache class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#headache">
    <!--  Headache is an individual of the Symptoms class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Nausea and Vomiting -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#nauseaOrVomitting">
        <!-- Nausea and Vomiting is a subclassification of Symptoms -->
```



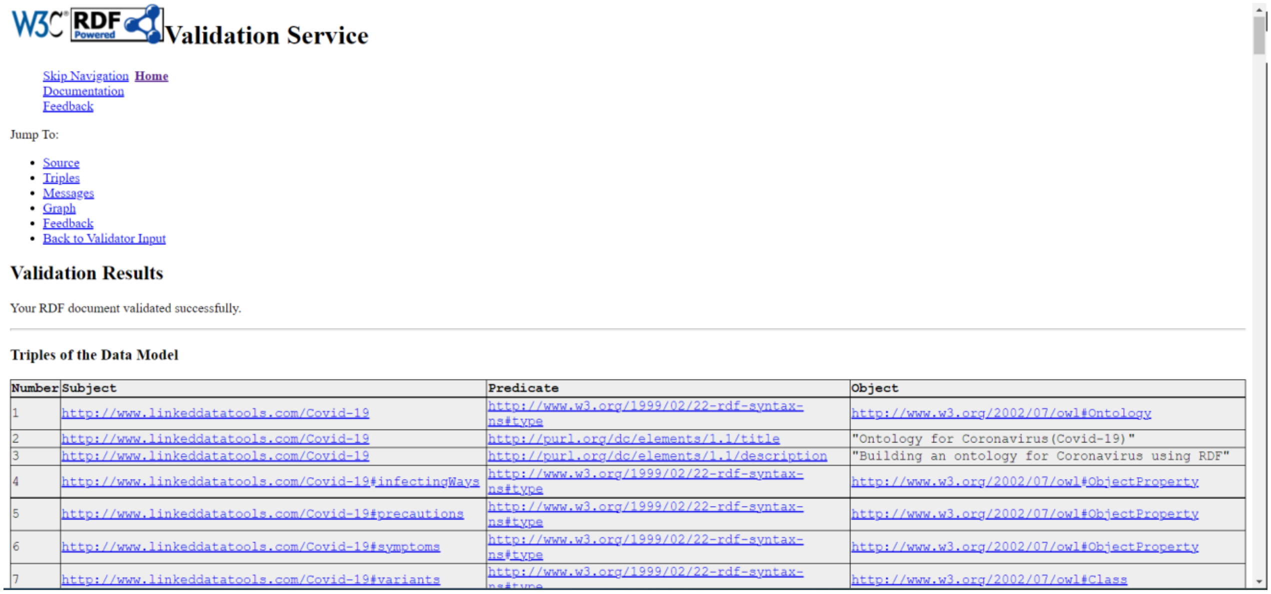
```
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
        <rdfs:label>Nausea and Vomiting</rdfs:label>
        <rdfs:comment>Nausea and Vomiting etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Muscle and body aches class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#nauseaOrVomitting">
    <!--  Muscle and body aches is an individual  of the Symptoms class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Muscle and body aches -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#muscleOrBodyAches">
        <!-- Muscle and body aches is a subclassification of Symptoms -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
        <rdfs:label>Muscle and body aches</rdfs:label>
        <rdfs:comment>Muscle and body aches etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Muscle and body aches class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#muscleOrBodyAches">
    <!--  Muscle and body aches is an individual  of the Symptoms class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - fatigue -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#fatigue">
        <!-- fatigue is a subclassification of Symptoms -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
        <rdfs:label>Fatigue</rdfs:label>
        <rdfs:comment>Fatigue etc</rdfs:comment>
    </owl:Class>

    <!--  Define the fatigue class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#fatigue">
    <!--  fatigue is an individual of the Symptoms class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#symptoms"/>
    </rdf:Description>

    <!-- Sub classes of precautions class starts from here -->
    <!-- OWL SubClass Definition - Getting vaccinated -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#getvaccinated">
        <!-- Getting vaccinated is a subclassification of Precautions -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
        <rdfs:label>Getting vaccinated</rdfs:label>
        <rdfs:comment>Getting vaccinated etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Getting vaccinated class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#getvaccinated">
    <!--  Getting vaccinated is an individual of the Precautions class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Wear face masks -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#wearingFaceMasks">
        <!-- Wear face masks is a subclassification of Precautions -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
        <rdfs:label>Wearing face masks</rdfs:label>
        <rdfs:comment>Wearing face masks etc</rdfs:comment>
    </owl:Class>

    <!--  Define the Wear face masks class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#wearingFaceMasks">
    <!--  Wear face masks is an individual  of the Precautions class  -->
```



```
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Wash hands frequently -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#washHandsFrequently">
        <!-- Masks is a subclassification of Precautions -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
        <rdfs:label>Wash hands frequently</rdfs:label>
        <rdfs:comment>Wash hands frequently</rdfs:comment>
    </owl:Class>

    <!--  Define the Clean Surface class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#washHandsFrequently">
    <!--  Masks is an individual of the Precautions class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Clean Surface -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#cleanSurface">
        <!-- Advocacy is a subclassification of Precautions -->
        <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
        <rdfs:label>Clean surface</rdfs:label>
        <rdfs:comment>Clean surface</rdfs:comment>
    </owl:Class>

    <!--  Define the Clean Surface class instance  -->
    <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#cleanSurface">
    <!--  Clean Surface is an individual of the Precautions class  -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
    </rdf:Description>

    <!-- OWL SubClass Definition - Keep distance -->
    <owl:Class rdf:about="http://www.linkeddatatools.com/Covid-19#keepDistance">
        <!-- Keep distance is a subclassification of Precautions -->
```

```
            <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
            <rdfs:label>Keep distance</rdfs:label>
            <rdfs:comment>Keep distance</rdfs:comment>
        </owl:Class>

        <!--  Define the Keep distance class instance  -->
        <rdf:Description rdf:about="http://www.linkeddatatools.com/Covid-19#keepDistance">
        <!--  Keep distance is an individual for the Precautions class  -->
        <rdf:type rdf:resource="http://www.linkeddatatools.com/Covid-19#precautions"/>
        </rdf:Description>

        </rdf:RDF>
```

Screenshots of the RDF document validation:



Jump To:

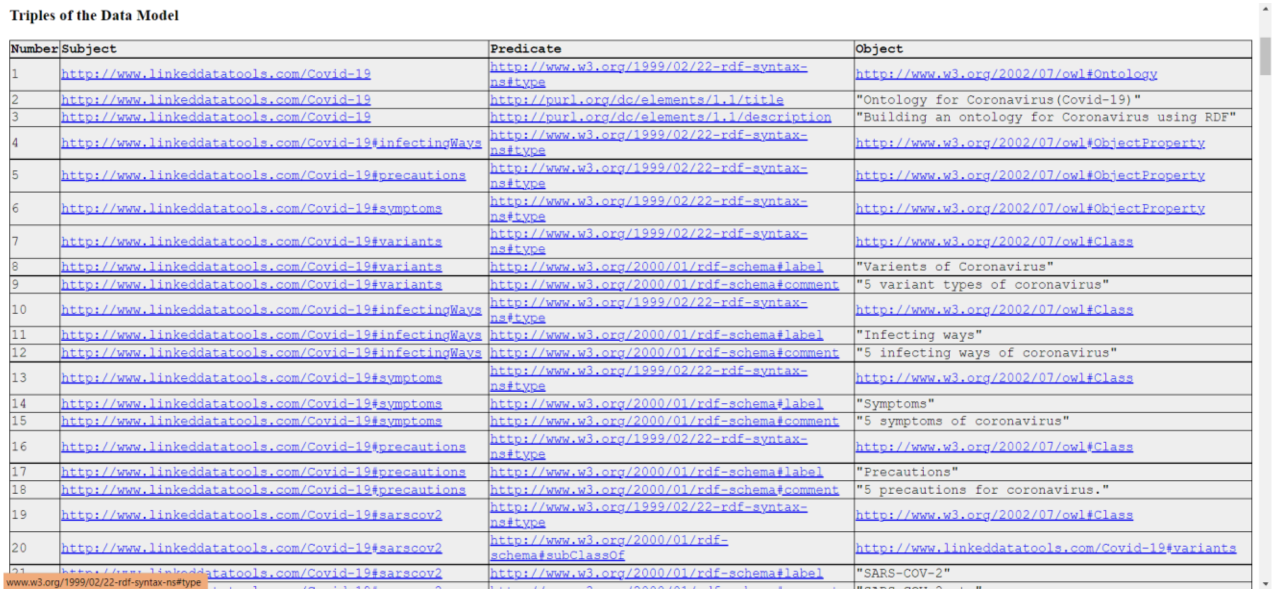
- Source
- Triples
- Messages
- Graph
- Feedback
- Back to Validator Input

Validation Results

Your RDF document validated successfully.

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://www.linkeddatatools.com/Covid-19	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
2	http://www.linkeddatatools.com/Covid-19	http://purl.org/dc/elements/1.1/title	"Ontology for Coronavirus(Covid-19)"
3	http://www.linkeddatatools.com/Covid-19	http://purl.org/dc/elements/1.1/description	"Building an ontology for Coronavirus using RDF"
4	http://www.linkeddatatools.com/Covid-19#infectingWays	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
5	http://www.linkeddatatools.com/Covid-19#precautions	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
6	http://www.linkeddatatools.com/Covid-19#symptoms	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
7	http://www.linkeddatatools.com/Covid-19#variants	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class

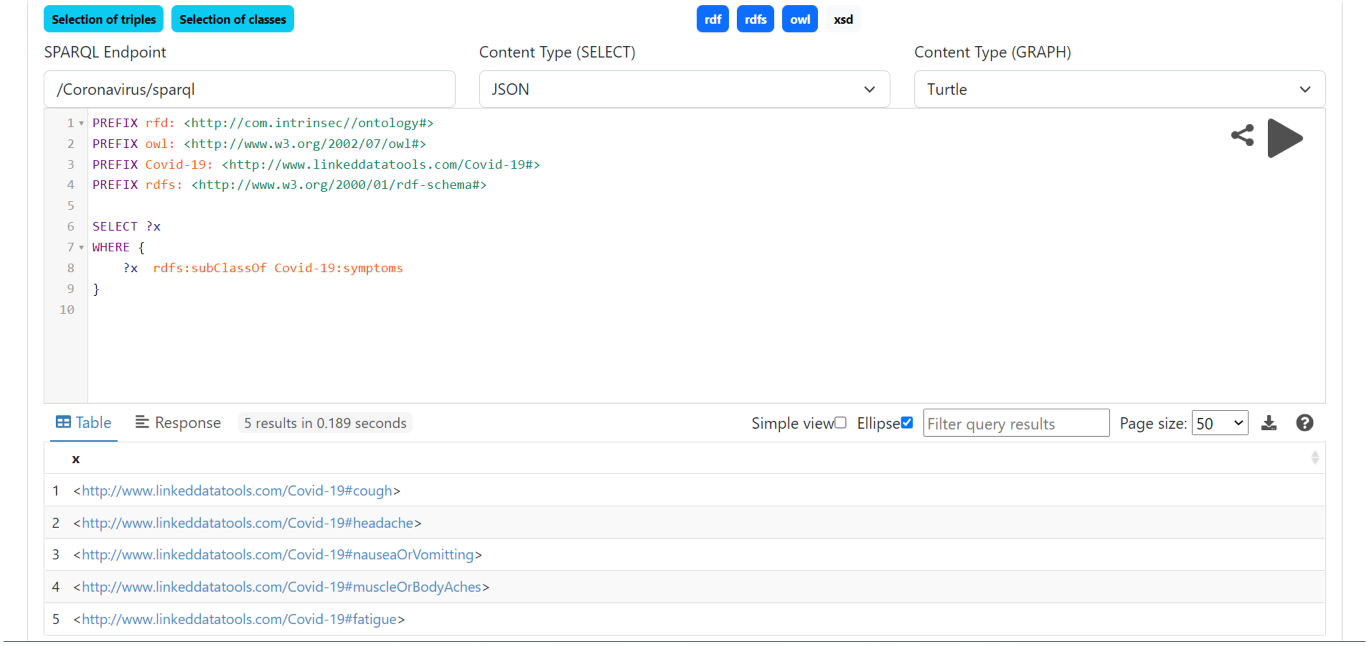


Triples of the Data Model

Number	Subject	Predicate	Object
1	http://www.linkeddatatools.com/Covid-19	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
2	http://www.linkeddatatools.com/Covid-19	http://purl.org/dc/elements/1.1/title	"Ontology for Coronavirus(Covid-19)"
3	http://www.linkeddatatools.com/Covid-19	http://purl.org/dc/elements/1.1/description	"Building an ontology for Coronavirus using RDF"
4	http://www.linkeddatatools.com/Covid-19#infectingWays	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
5	http://www.linkeddatatools.com/Covid-19#precautions	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
6	http://www.linkeddatatools.com/Covid-19#symptoms	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
7	http://www.linkeddatatools.com/Covid-19#variants	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
8	http://www.linkeddatatools.com/Covid-19#variants	http://www.w3.org/2000/01/rdf-schema#label	"Varients of Coronavirus"
9	http://www.linkeddatatools.com/Covid-19#variants	http://www.w3.org/2000/01/rdf-schema#comment	"5 variant types of coronavirus"
10	http://www.linkeddatatools.com/Covid-19#infectingWays	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
11	http://www.linkeddatatools.com/Covid-19#infectingWays	http://www.w3.org/2000/01/rdf-schema#label	"Infecting ways"
12	http://www.linkeddatatools.com/Covid-19#infectingWays	http://www.w3.org/2000/01/rdf-schema#comment	"5 infecting ways of coronavirus"
13	http://www.linkeddatatools.com/Covid-19#symptoms	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
14	http://www.linkeddatatools.com/Covid-19#symptoms	http://www.w3.org/2000/01/rdf-schema#label	"Symptoms"
15	http://www.linkeddatatools.com/Covid-19#symptoms	http://www.w3.org/2000/01/rdf-schema#comment	"5 symptoms of coronavirus"
16	http://www.linkeddatatools.com/Covid-19#precautions	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
17	http://www.linkeddatatools.com/Covid-19#precautions	http://www.w3.org/2000/01/rdf-schema#label	"Precautions"
18	http://www.linkeddatatools.com/Covid-19#precautions	http://www.w3.org/2000/01/rdf-schema#comment	"5 precautions for coronavirus."
19	http://www.linkeddatatools.com/Covid-19#sarscov2	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
20	http://www.linkeddatatools.com/Covid-19#sarscov2	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/Covid-19#variants
21	http://www.linkeddatatools.com/Covid-19#sarscov2	http://www.w3.org/2000/01/rdf-schema#label	"SARS-COV-2"

4. Queries for covid-19 ontology :

```
PREFIX rfd: <http://com.intrinsec//ontology#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>


SELECT ?x

WHERE {

        ?x        rdfs:subClassOf Covid-19:symptoms

}
```



SPARQL Endpoint: /Coronavirus/sparql

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

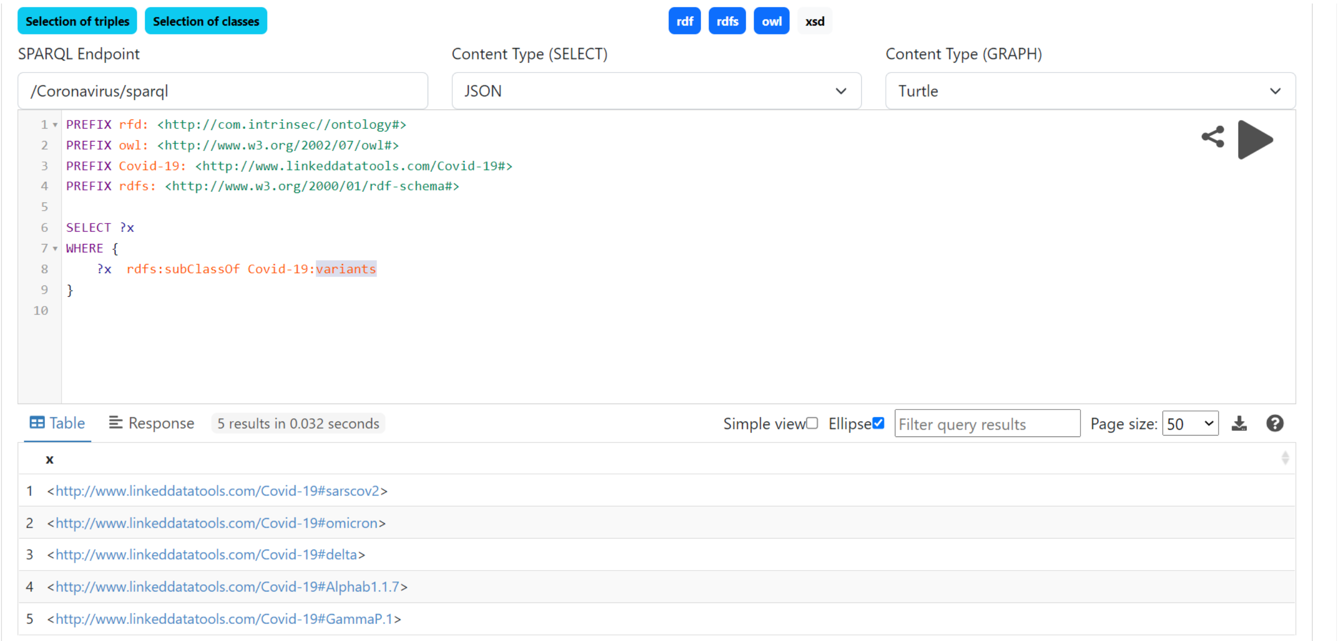
```
PREFIX rfd: <http://com.intrinsec//ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x
WHERE {
    ?x  rdfs:subClassOf Covid-19:symptoms
}
```

5 results in 0.189 seconds

Simple view□ Ellipse☑ Page size: 50

	x
1	<http://www.linkeddatatools.com/Covid-19#cough>
2	<http://www.linkeddatatools.com/Covid-19#headache>
3	<http://www.linkeddatatools.com/Covid-19#nauseaOrVomitting>
4	<http://www.linkeddatatools.com/Covid-19#muscleOrBodyAches>
5	<http://www.linkeddatatools.com/Covid-19#fatigue>

```
PREFIX rfd: <http://com.intrinsec//ontology#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>


SELECT ?x

WHERE {

        ?x      rdfs:subClassOf Covid-19:variants

}
```



SPARQL Endpoint: /Coronavirus/sparql

Content Type (SELECT): JSON

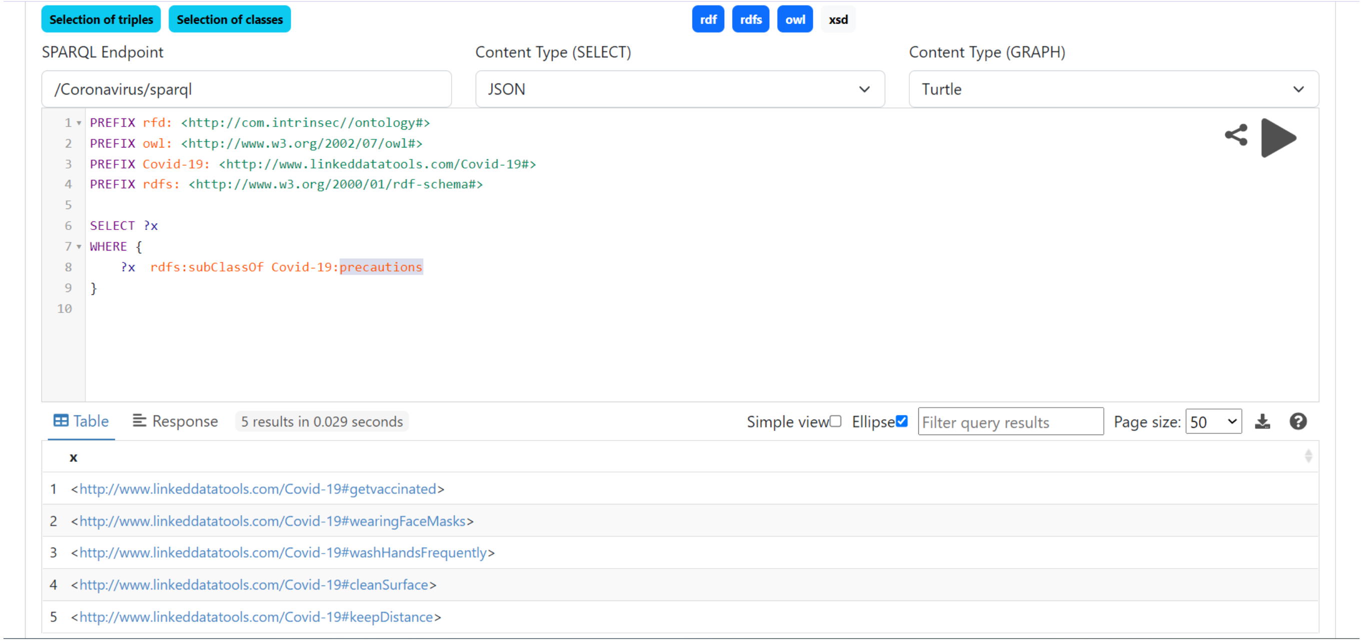
Content Type (GRAPH): Turtle

```
PREFIX rfd: <http://com.intrinsec//ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x
WHERE {
    ?x  rdfs:subClassOf Covid-19:variants
}
```

5 results in 0.032 seconds

	x
1	<http://www.linkeddatatools.com/Covid-19#sarscov2>
2	<http://www.linkeddatatools.com/Covid-19#omicron>
3	<http://www.linkeddatatools.com/Covid-19#delta>
4	<http://www.linkeddatatools.com/Covid-19#Alphab1.1.7>
5	<http://www.linkeddatatools.com/Covid-19#GammaP.1>

```
PREFIX rfd: <http://com.intrinsec//ontology#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>


SELECT ?x

WHERE {

        ?x      rdfs:subClassOf Covid-19:precautions

}
```



SPARQL Endpoint: /Coronavirus/sparql

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

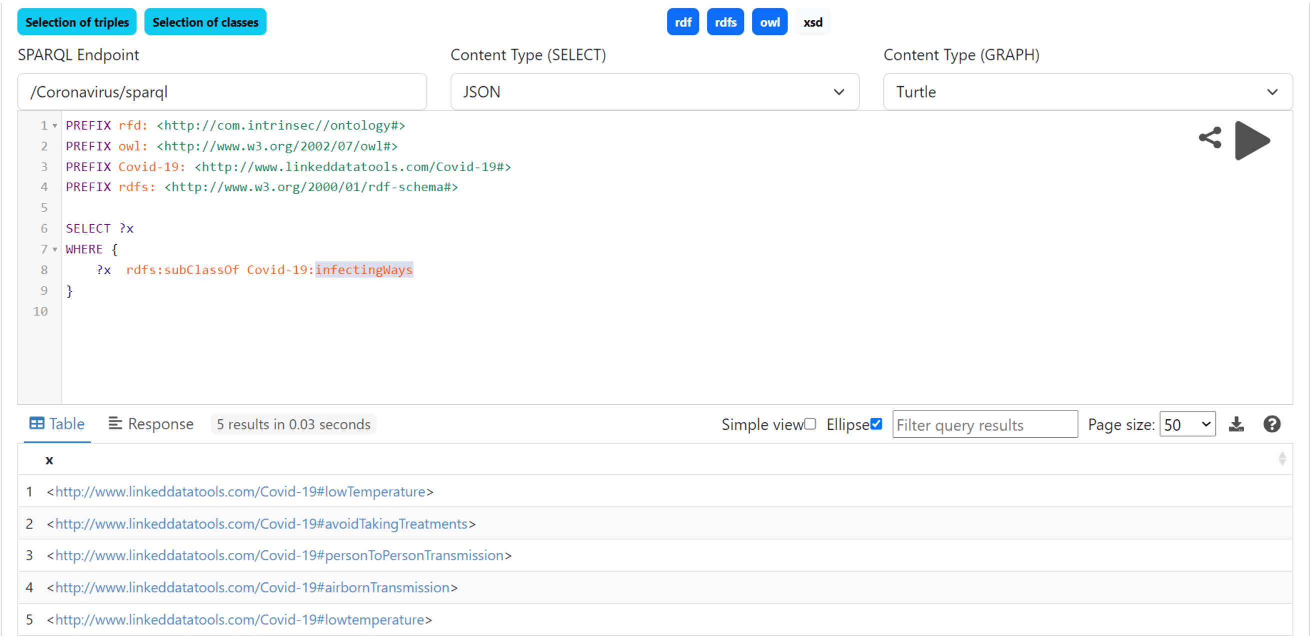
```
PREFIX rfd: <http://com.intrinsec//ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x
WHERE {
    ?x  rdfs:subClassOf Covid-19:precautions
}
```

5 results in 0.029 seconds

Page size: 50

	x
1	<http://www.linkeddatatools.com/Covid-19#getvaccinated>
2	<http://www.linkeddatatools.com/Covid-19#wearingFaceMasks>
3	<http://www.linkeddatatools.com/Covid-19#washHandsFrequently>
4	<http://www.linkeddatatools.com/Covid-19#cleanSurface>
5	<http://www.linkeddatatools.com/Covid-19#keepDistance>

```
PREFIX rfd: <http://com.intrinsec//ontology#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>


SELECT ?x

WHERE {

   ?x rdfs:subClassOf Covid-19:variants .

   ?x rdfs:subClassOf [

                          a owl:Restriction ;

                          owl:infectingWays Covid-19:lowTemperature;

             ]

}
```



SPARQL Endpoint: /Coronavirus/sparql

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

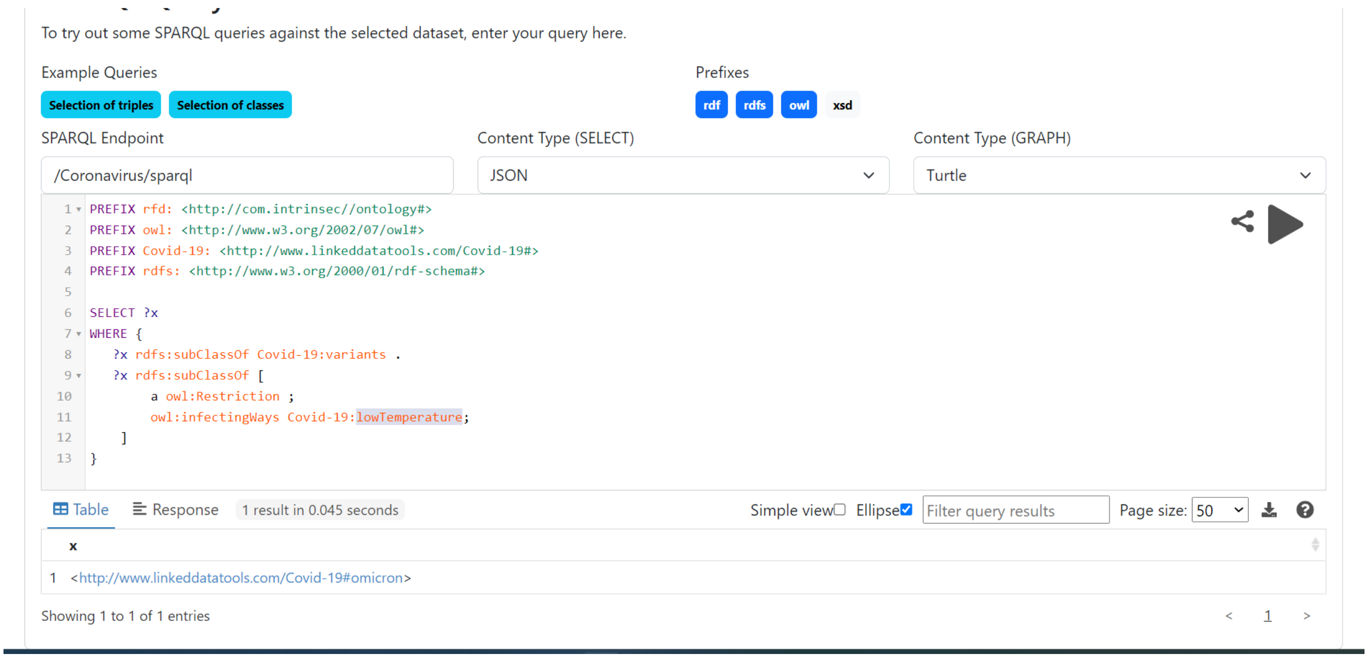
```
PREFIX rfd: <http://com.intrinsec//ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x
WHERE {
    ?x  rdfs:subClassOf Covid-19:infectingWays
}

```

5 results in 0.03 seconds

Simple view☐ Ellipse☑ Filter query results Page size: 50

	x
1	<http://www.linkeddatatools.com/Covid-19#lowTemperature>
2	<http://www.linkeddatatools.com/Covid-19#avoidTakingTreatments>
3	<http://www.linkeddatatools.com/Covid-19#personToPersonTransmission>
4	<http://www.linkeddatatools.com/Covid-19#airbornTransmission>
5	<http://www.linkeddatatools.com/Covid-19#lowtemperature>

```
PREFIX rfd: <http://com.intrinsec//ontology#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>


SELECT ?x

WHERE {

  ?x rdfs:subClassOf Covid-19:variants .

  ?x rdfs:subClassOf [

                    a owl:Restriction ;

                    owl:someValuesFrom Covid-19:infectingWays;

          ]

}
```



To try out some SPARQL queries against the selected dataset, enter your query here.

Example Queries

Prefixes

SPARQL Endpoint: /Coronavirus/sparql

Content Type (SELECT): JSON

Content Type (GRAPH): Turtle

```
PREFIX rfd: <http://com.intrinsec//ontology#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Covid-19: <http://www.linkeddatatools.com/Covid-19#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x
WHERE {
   ?x rdfs:subClassOf Covid-19:variants .
   ?x rdfs:subClassOf [
        a owl:Restriction ;
        owl:infectingWays Covid-19:lowTemperature;
     ]
}
```

1 result in 0.045 seconds

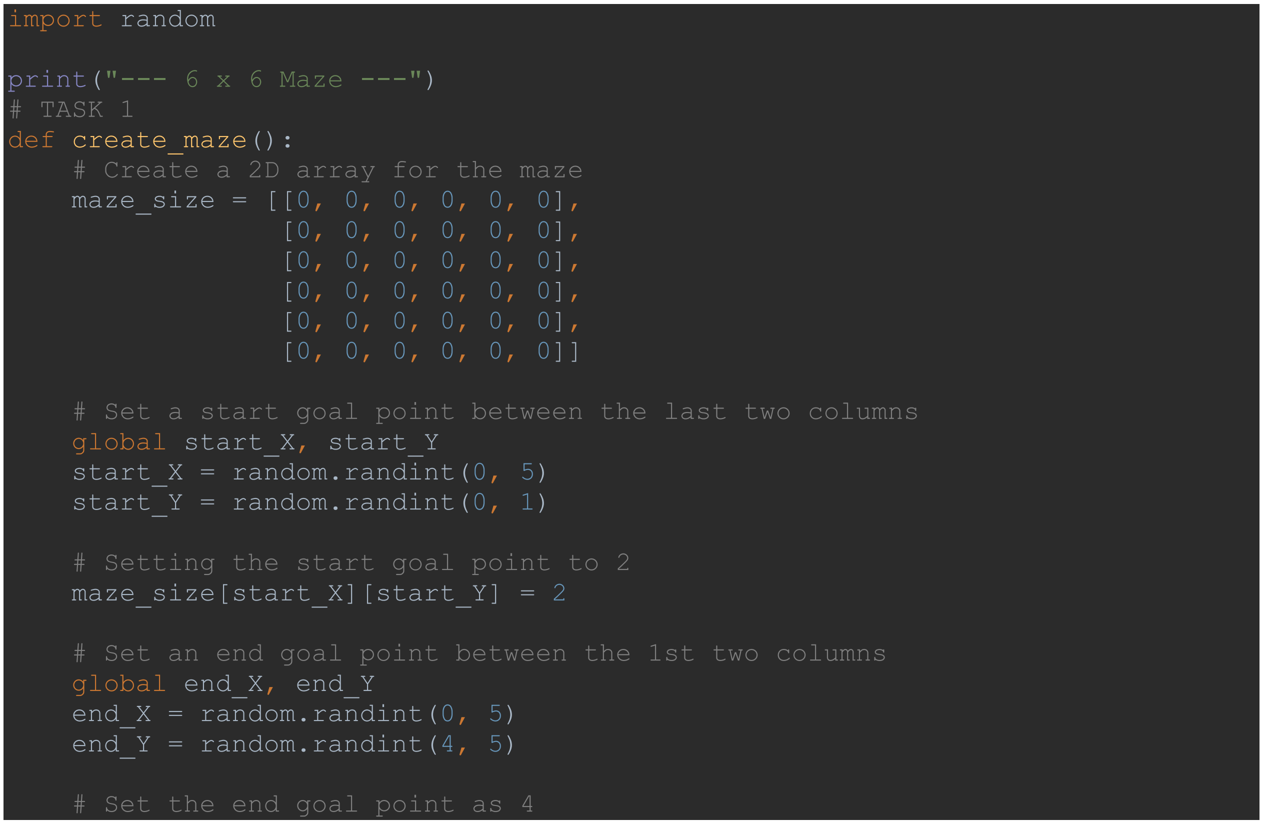
Simple view☐ Ellipse☑ Filter query results Page size: 50

	x
1	<http://www.linkeddatatools.com/Covid-19#omicron>

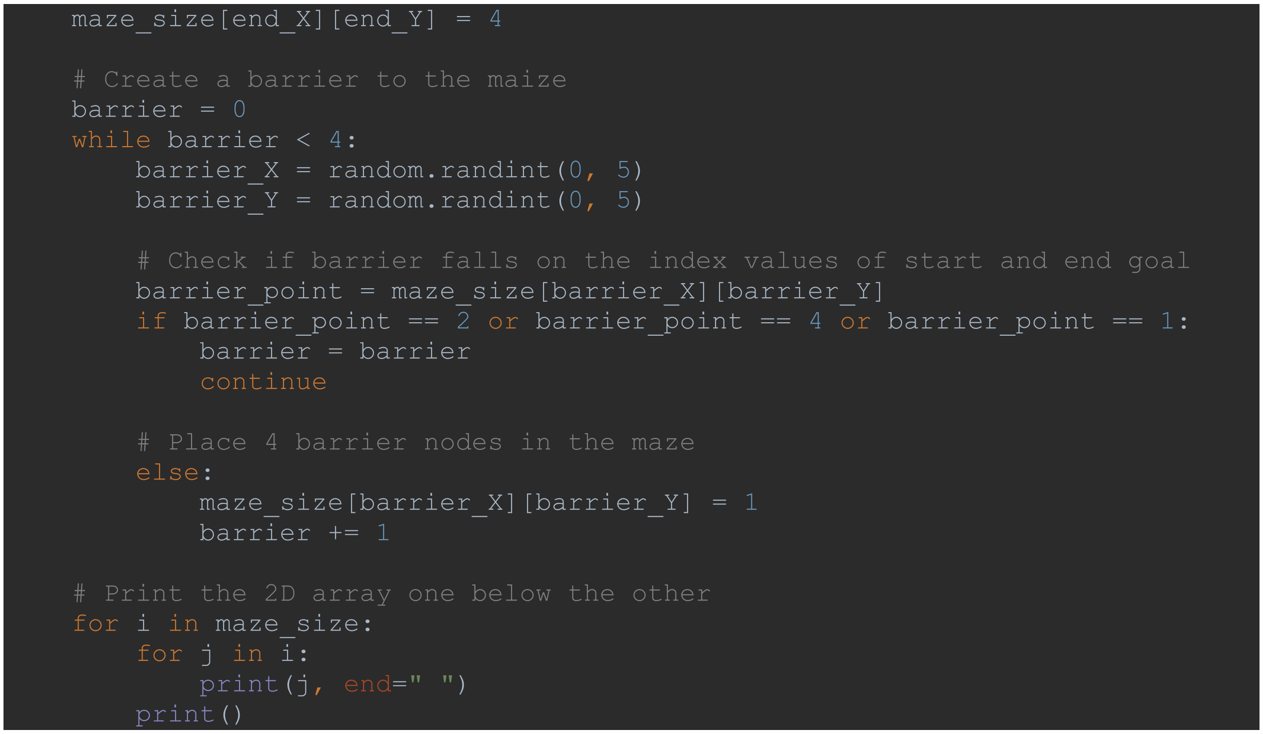
Showing 1 to 1 of 1 entries

Question 03

Task 1

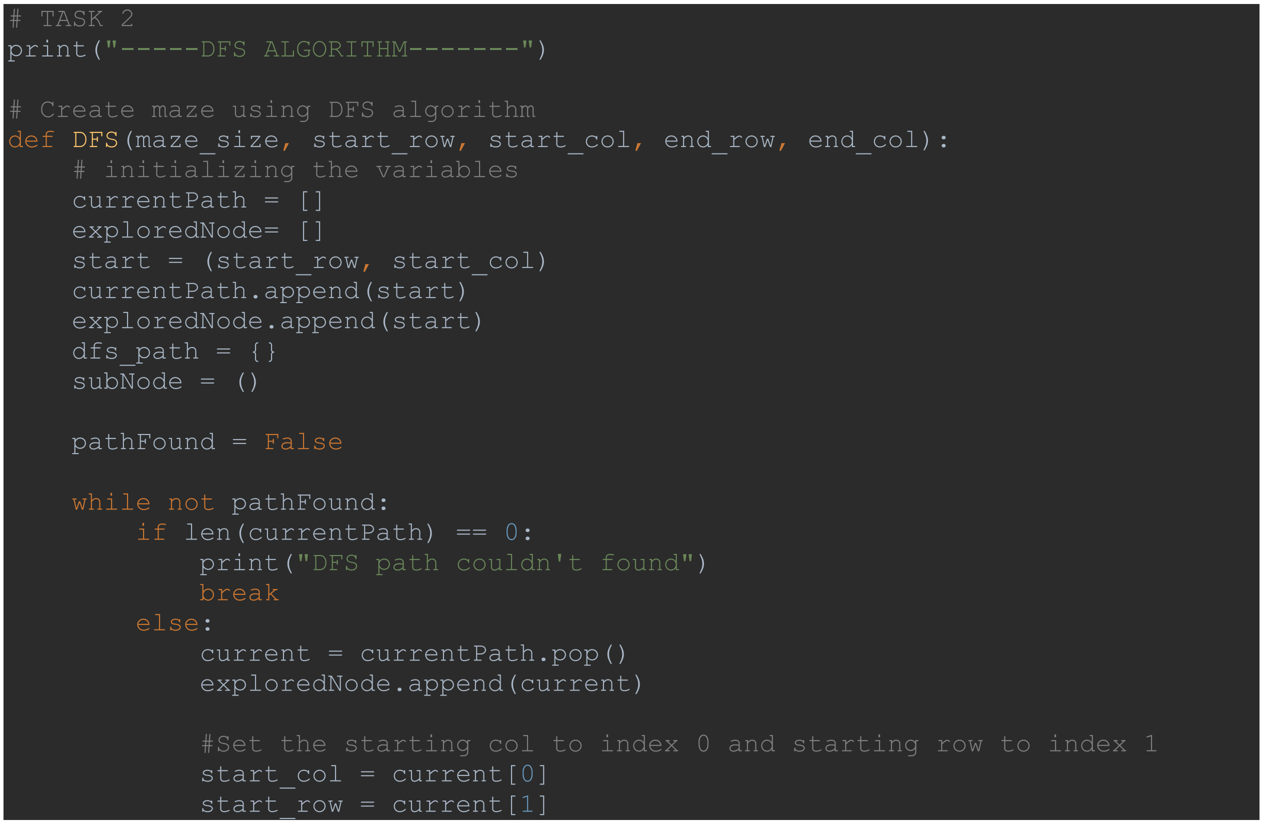


```
import random

print("--- 6 x 6 Maze ---")
# TASK 1
def create_maze():
    # Create a 2D array for the maze
    maze_size = [[0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0]]

    # Set a start goal point between the last two columns
    global start_X, start_Y
    start_X = random.randint(0, 5)
    start_Y = random.randint(0, 1)

    # Setting the start goal point to 2
    maze_size[start_X][start_Y] = 2

    # Set an end goal point between the 1st two columns
    global end_X, end_Y
    end_X = random.randint(0, 5)
    end_Y = random.randint(4, 5)

    # Set the end goal point as 4
```

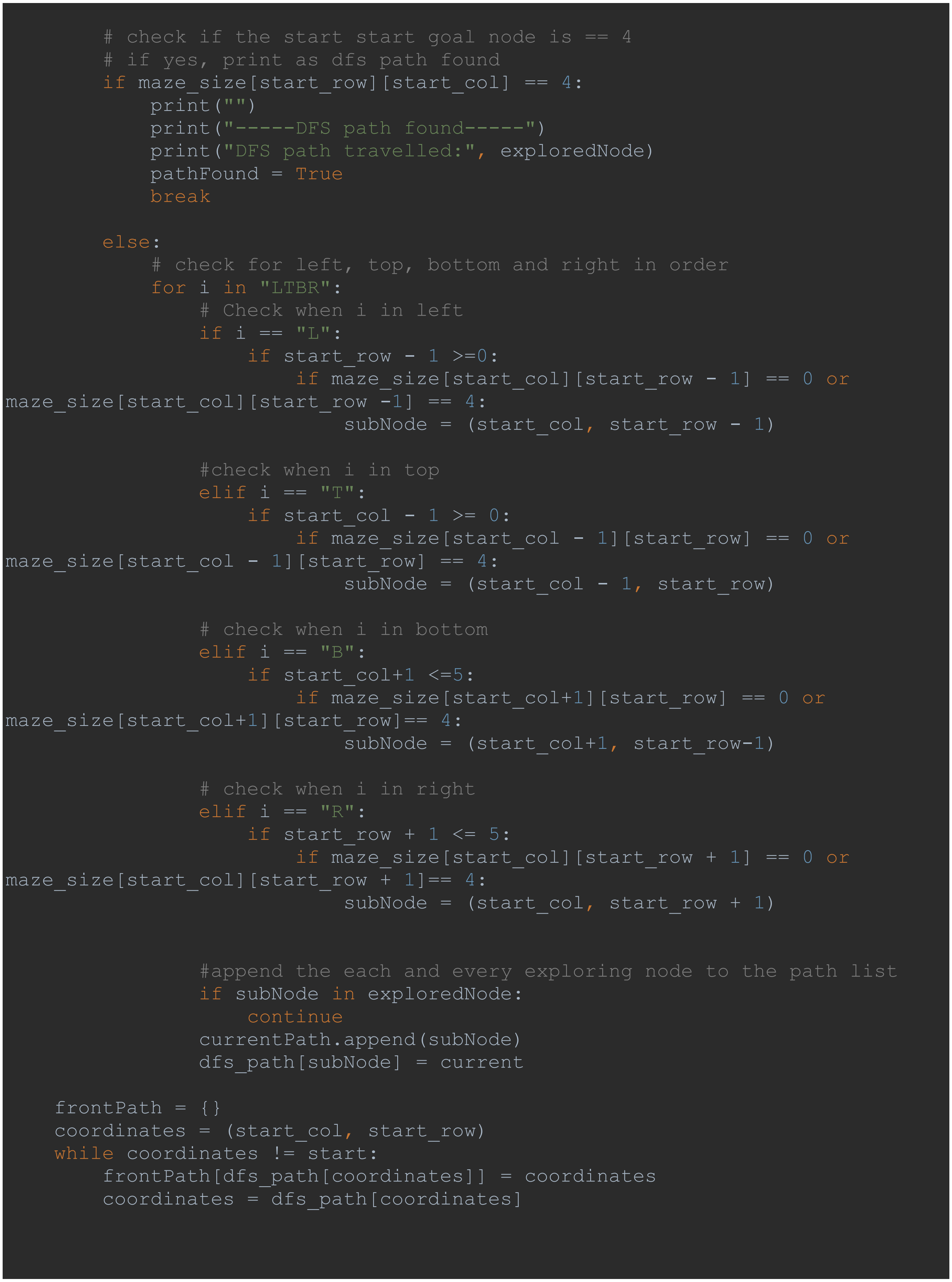


```
    maze_size[end_X][end_Y] = 4

    # Create a barrier to the maize
    barrier = 0
    while barrier < 4:
        barrier_X = random.randint(0, 5)
        barrier_Y = random.randint(0, 5)

        # Check if barrier falls on the index values of start and end goal
        barrier_point = maze_size[barrier_X][barrier_Y]
        if barrier_point == 2 or barrier_point == 4 or barrier_point == 1:
            barrier = barrier
            continue

        # Place 4 barrier nodes in the maze
        else:
            maze_size[barrier_X][barrier_Y] = 1
            barrier += 1

    # Print the 2D array one below the other
    for i in maze_size:
        for j in i:
            print(j, end=" ")
        print()
```

Task 2



```
# TASK 2
print("-----DFS ALGORITHM-------")

# Create maze using DFS algorithm
def DFS(maze_size, start_row, start_col, end_row, end_col):
    # initializing the variables
    currentPath = []
    exploredNode= []
    start = (start_row, start_col)
    currentPath.append(start)
    exploredNode.append(start)
    dfs_path = {}
    subNode = ()

    pathFound = False

    while not pathFound:
        if len(currentPath) == 0:
            print("DFS path couldn't found")
            break
        else:
            current = currentPath.pop()
            exploredNode.append(current)

            #Set the starting col to index 0 and starting row to index 1
            start_col = current[0]
            start_row = current[1]
```



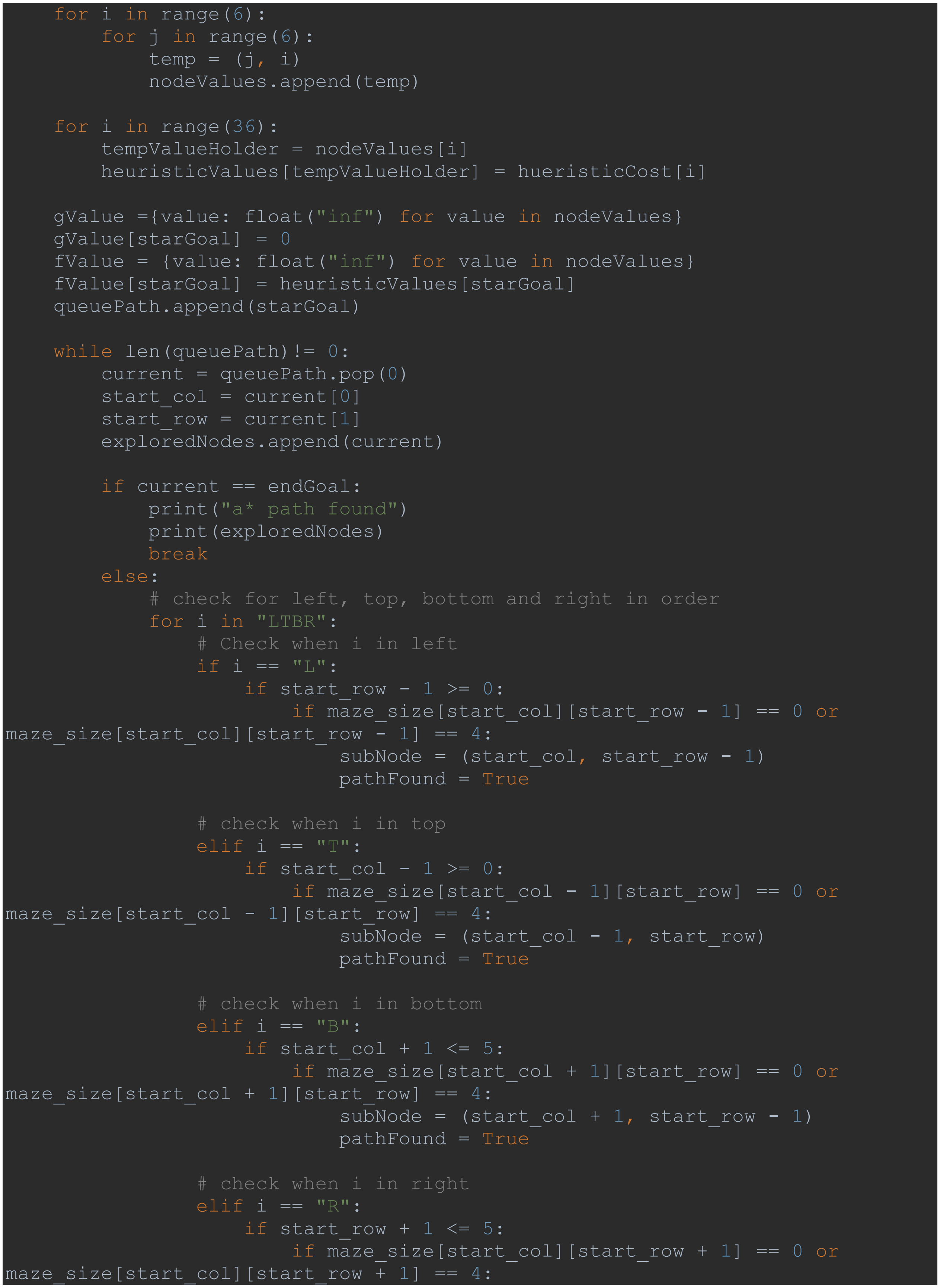
```
        # check if the start start goal node is == 4
        # if yes, print as dfs path found
        if maze_size[start_row][start_col] == 4:
            print("")
            print("-----DFS path found-----")
            print("DFS path travelled:", exploredNode)
            pathFound = True
            break

        else:
            # check for left, top, bottom and right in order
            for i in "LTBR":
                # Check when i in left
                if i == "L":
                    if start_row - 1 >=0:
                        if maze_size[start_col][start_row - 1] == 0 or
maze_size[start_col][start_row -1] == 4:
                            subNode = (start_col, start_row - 1)

                #check when i in top
                elif i == "T":
                    if start_col - 1 >= 0:
                        if maze_size[start_col - 1][start_row] == 0 or
maze_size[start_col - 1][start_row] == 4:
                            subNode = (start_col - 1, start_row)

                # check when i in bottom
                elif i == "B":
                    if start_col+1 <=5:
                        if maze_size[start_col+1][start_row] == 0 or
maze_size[start_col+1][start_row]== 4:
                            subNode = (start_col+1, start_row-1)

                # check when i in right
                elif i == "R":
                    if start_row + 1 <= 5:
                        if maze_size[start_col][start_row + 1] == 0 or
maze_size[start_col][start_row + 1]== 4:
                            subNode = (start_col, start_row + 1)


                #append the each and every exploring node to the path list
                if subNode in exploredNode:
                    continue
                currentPath.append(subNode)
                dfs_path[subNode] = current

    frontPath = {}
    coordinates = (start_col, start_row)
    while coordinates != start:
        frontPath[dfs_path[coordinates]] = coordinates
        coordinates = dfs_path[coordinates]
    #Calculate the time taken to travel the path using DFS algorithm
    if pathFound == True:
        timeTaken = len(dfs_path)
        print("Time taken by DFS to find the goal node: "+ str(timeTaken)+ " minutes")
        print("")
```

Task 3

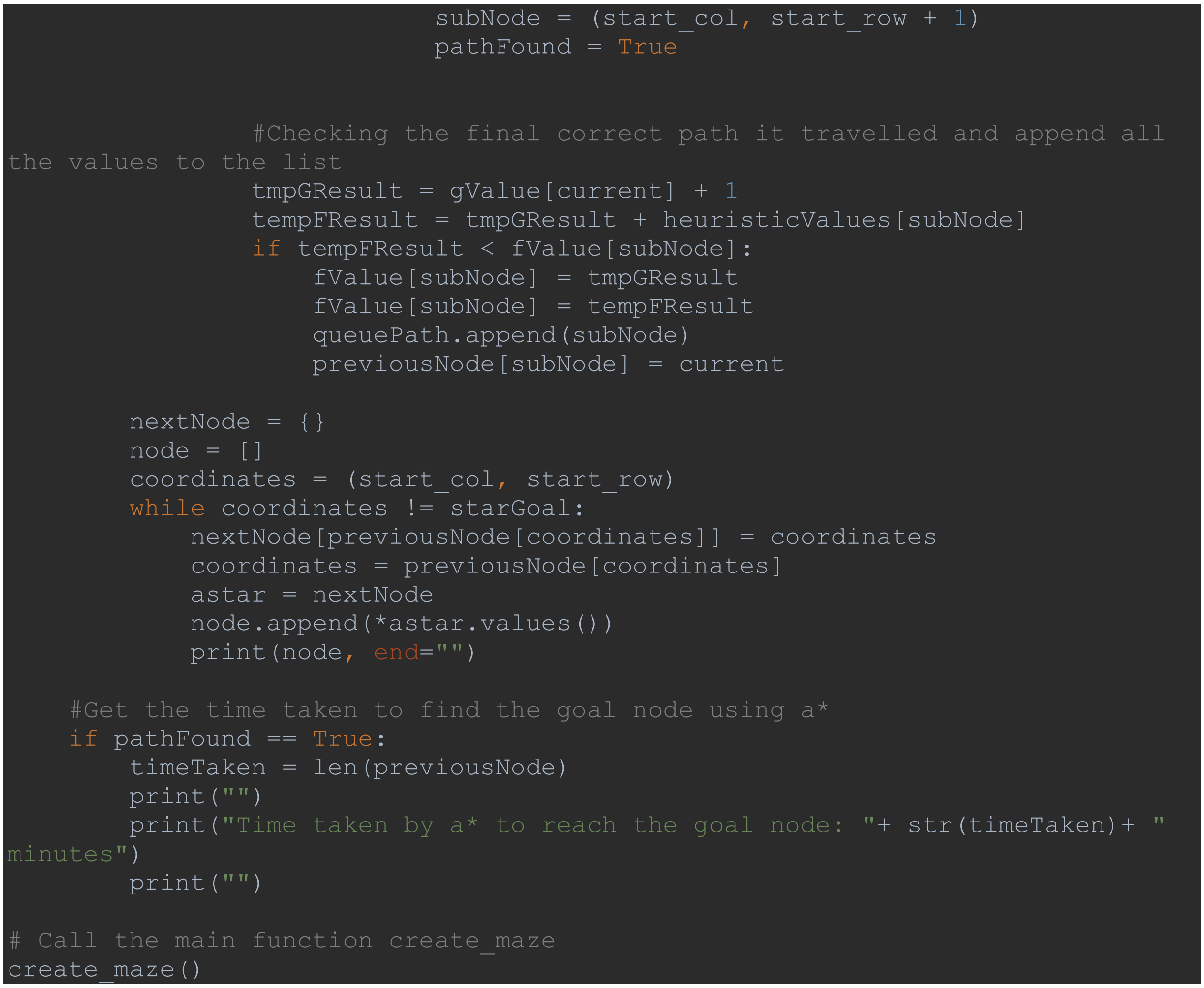
```
# Task 3
# Find the heuristic cost value
def getHeuristicCost(end_X, end_Y):
    a = 0
    gx = end_X
    gy = end_Y
    heuristicList = []

    while a < 6:
        b = 0
        while b < 6:
            heuristicValue = max(abs(a - gx), abs(b - gy))
            heuristicList.append(heuristicValue)
            b += 1
        a += 1
    return heuristicList
```

Task 4

```
# Task 4
#Cretae a* algorithm for maze
def aStar(maze_size, start_row, start_col, end_row, end_col):
    #Initialize the variables
    starGoal = (start_col, start_row)
    endGoal = (end_col, end_row)
    queuePath = []
    nodeValues = []
    exploredNodes = []
    heuristicValues = {}
    subNode = starGoal
    previousNode = {}

    pathFound = False

    #get the gx,gy values passing from hueristic cost function
    hueristicCost = getHeuristicCost(end_row, end_col)
    print("------a* algorithm------")
    print("Heuristic cost: ", hueristicCost)
    print("")
    print("a* path travelled: ")
```



```
    for i in range(6):
        for j in range(6):
            temp = (j, i)
            nodeValues.append(temp)

    for i in range(36):
        tempValueHolder = nodeValues[i]
        heuristicValues[tempValueHolder] = hueristicCost[i]

    gValue ={value: float("inf") for value in nodeValues}
    gValue[starGoal] = 0
    fValue = {value: float("inf") for value in nodeValues}
    fValue[starGoal] = heuristicValues[starGoal]
    queuePath.append(starGoal)

    while len(queuePath)!= 0:
        current = queuePath.pop(0)
        start_col = current[0]
        start_row = current[1]
        exploredNodes.append(current)

        if current == endGoal:
            print("a* path found")
            print(exploredNodes)
            break
        else:
            # check for left, top, bottom and right in order
            for i in "LTBR":
                # Check when i in left
                if i == "L":
                    if start_row - 1 >= 0:
                        if maze_size[start_col][start_row - 1] == 0 or
maze_size[start_col][start_row - 1] == 4:
                            subNode = (start_col, start_row - 1)
                            pathFound = True

                # check when i in top
                elif i == "T":
                    if start_col - 1 >= 0:
                        if maze_size[start_col - 1][start_row] == 0 or
maze_size[start_col - 1][start_row] == 4:
                            subNode = (start_col - 1, start_row)
                            pathFound = True

                # check when i in bottom
                elif i == "B":
                    if start_col + 1 <= 5:
                        if maze_size[start_col + 1][start_row] == 0 or
maze_size[start_col + 1][start_row] == 4:
                            subNode = (start_col + 1, start_row - 1)
                            pathFound = True

                # check when i in right
                elif i == "R":
                    if start_row + 1 <= 5:
                        if maze_size[start_col][start_row + 1] == 0 or
maze_size[start_col][start_row + 1] == 4:
```

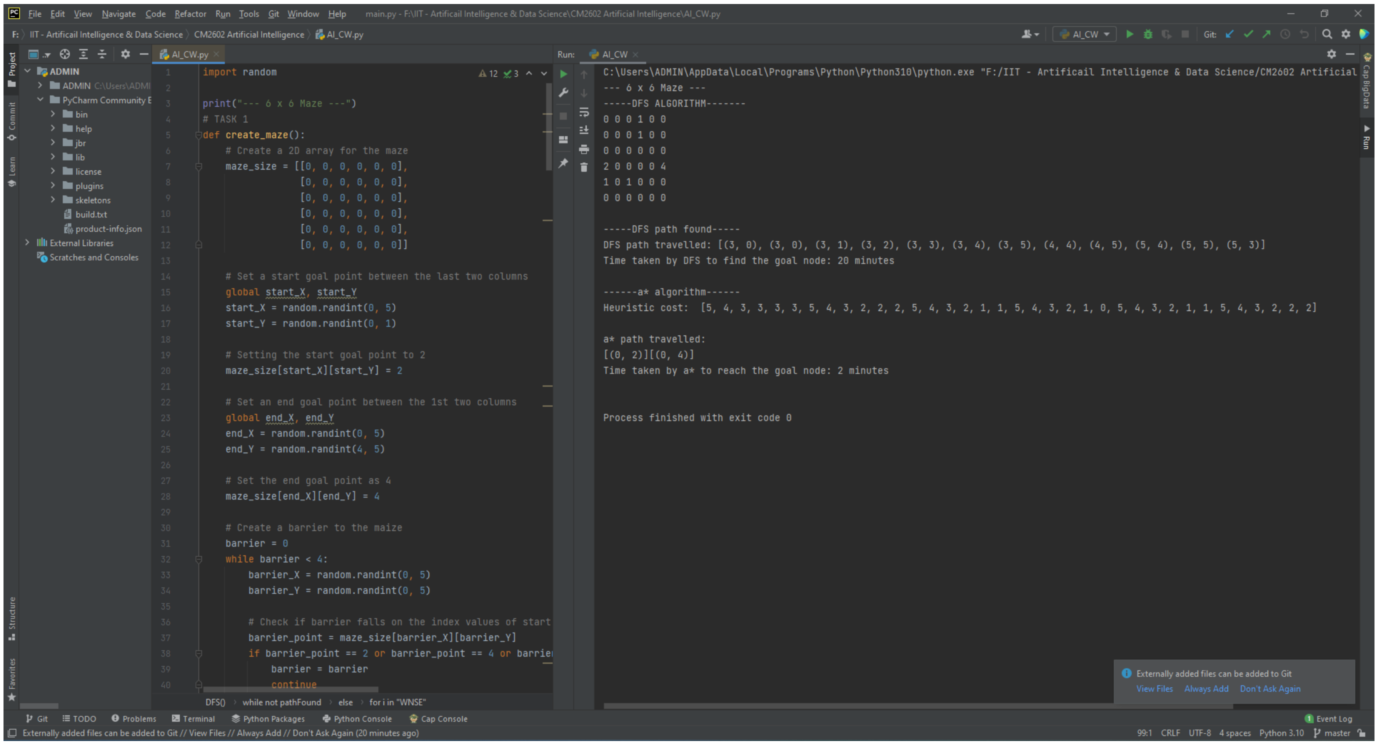


```
                            subNode = (start_col, start_row + 1)
                            pathFound = True


                #Checking the final correct path it travelled and append all
the values to the list
                tmpGResult = gValue[current] + 1
                tempFResult = tmpGResult + heuristicValues[subNode]
                if tempFResult < fValue[subNode]:
                    fValue[subNode] = tmpGResult
                    fValue[subNode] = tempFResult
                    queuePath.append(subNode)
                    previousNode[subNode] = current

        nextNode = {}
        node = []
        coordinates = (start_col, start_row)
        while coordinates != starGoal:
            nextNode[previousNode[coordinates]] = coordinates
            coordinates = previousNode[coordinates]
            astar = nextNode
            node.append(*astar.values())
            print(node, end="")

    #Get the time taken to find the goal node using a*
    if pathFound == True:
        timeTaken = len(previousNode)
        print("")
        print("Time taken by a* to reach the goal node: "+ str(timeTaken)+ "
minutes")
        print("")

# Call the main function create_maze
create_maze()
```

Task 5

Screenshots of the outputs:

Output 1:

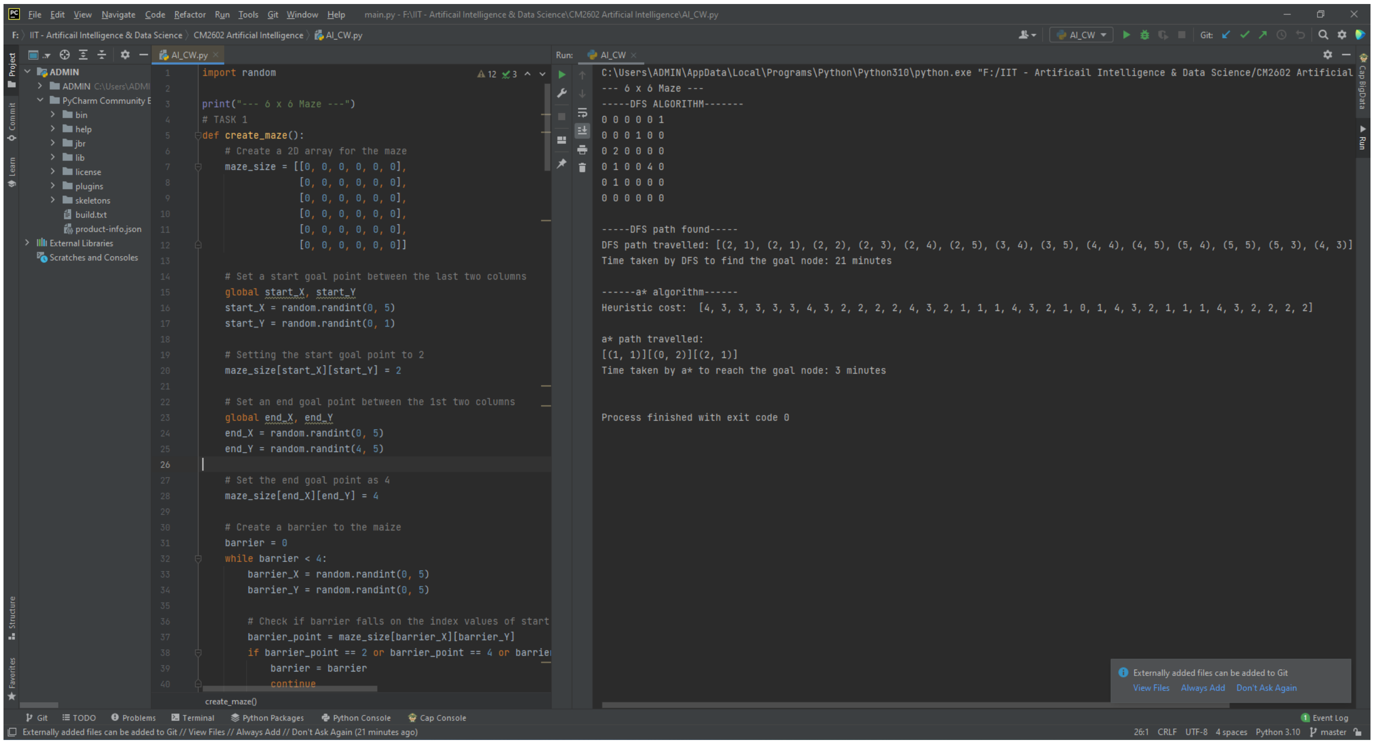


```python
import random

print("--- 6 x 6 Maze ---")
# TASK 1
def create_maze():
    # Create a 2D array for the maze
    maze_size = [[0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0]]

    # Set a start goal point between the last two columns
    global start_X, start_Y
    start_X = random.randint(0, 5)
    start_Y = random.randint(0, 1)

    # Setting the start goal point to 2
    maze_size[start_X][start_Y] = 2

    # Set an end goal point between the 1st two columns
    global end_X, end_Y
    end_X = random.randint(0, 5)
    end_Y = random.randint(4, 5)

    # Set the end goal point as 4
    maze_size[end_X][end_Y] = 4

    # Create a barrier to the maize
    barrier = 0
    while barrier < 4:
        barrier_X = random.randint(0, 5)
        barrier_Y = random.randint(0, 5)

        # Check if barrier falls on the index values of start
        barrier_point = maze_size[barrier_X][barrier_Y]
        if barrier_point == 2 or barrier_point == 4 or barrie
            barrier = barrier
            continue
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\python.exe "F:/IIT - Artificail Intelligence & Data Science/CM2602 Artificial
--- 6 x 6 Maze ---
-----DFS ALGORITHM-------
0 0 0 1 0 0
0 0 0 1 0 0
0 0 0 0 0 0
2 0 0 0 0 4
1 0 1 0 0 0
0 0 0 0 0 0

-----DFS path found-----
DFS path travelled: [(3, 0), (3, 0), (3, 1), (3, 2), (3, 3), (3, 4), (3, 5), (4, 4), (4, 5), (5, 4), (5, 5), (5, 3)]
Time taken by DFS to find the goal node: 20 minutes

------a* algorithm------
Heuristic cost:  [5, 4, 3, 3, 3, 3, 5, 4, 3, 2, 2, 2, 5, 4, 3, 2, 1, 1, 5, 4, 3, 2, 1, 0, 5, 4, 3, 2, 1, 1, 5, 4, 3, 2, 2, 2]

a* path travelled:
[(0, 2)][(0, 4)]
Time taken by a* to reach the goal node: 2 minutes


Process finished with exit code 0
```

File Edit View Navigate Code Refactor Run Tools Git Window Help
main.py - F:\IIT - Artificail Intelligence & Data Science\CM2602 Artificial Intelligence\AI_CW.py
F: IIT - Artificail Intelligence & Data Science CM2602 Artificial Intelligence AI_CW.py
ADMIN
ADMIN C:\Users\ADMI
PyCharm Community E
bin
help
jbr
lib
license
plugins
skeletons
build.txt
product-info.json
External Libraries
Scratches and Consoles
AI_CW.py
Run: AI_CW
DFS() › while not pathFound › else › for i in "WNSE"
Git TODO Problems Terminal Python Packages Python Console Cap Console
Externally added files can be added to Git
View Files Always Add Don't Ask Again
Event Log
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (20 minutes ago)
99:1 CRLF UTF-8 4 spaces Python 3.10 master

Output 2:



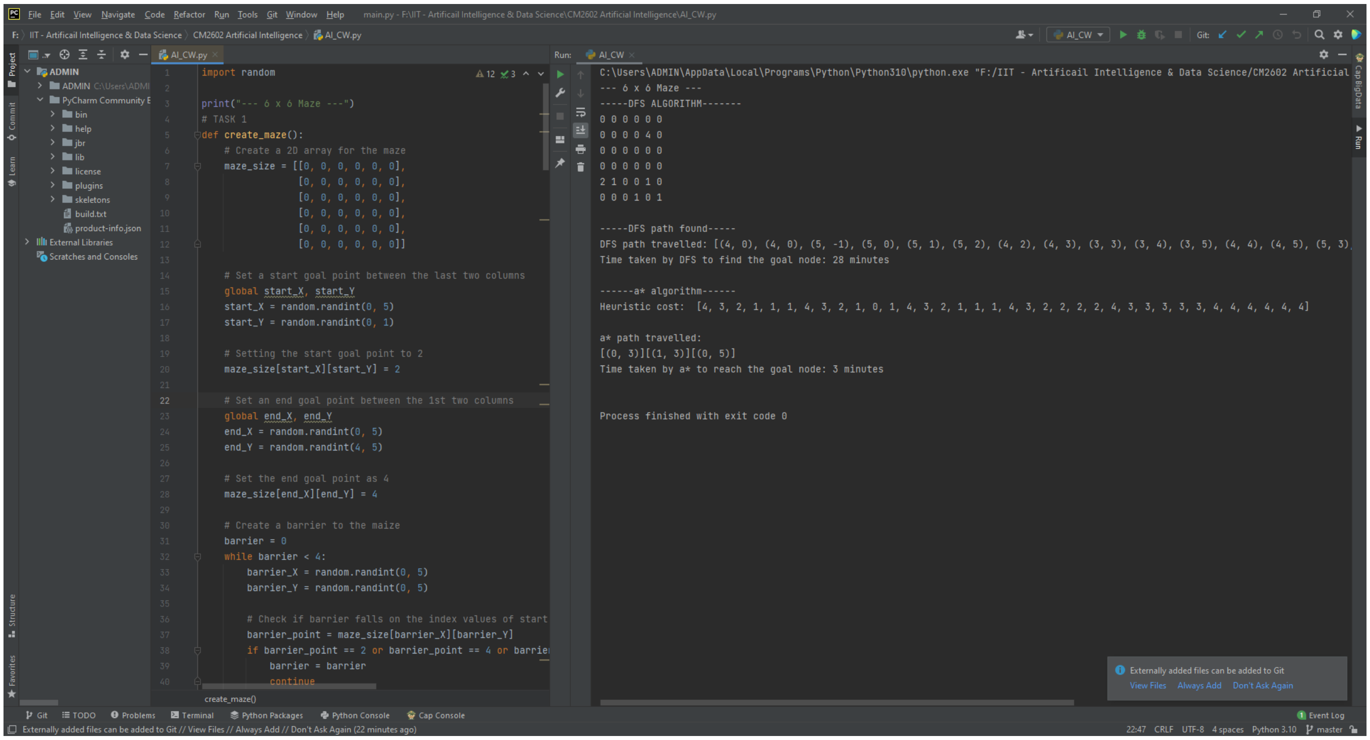
```
import random

print("--- 6 x 6 Maze ---")
# TASK 1
def create_maze():
    # Create a 2D array for the maze
    maze_size = [[0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0]]

    # Set a start goal point between the last two columns
    global start_X, start_Y
    start_X = random.randint(0, 5)
    start_Y = random.randint(0, 1)

    # Setting the start goal point to 2
    maze_size[start_X][start_Y] = 2

    # Set an end goal point between the 1st two columns
    global end_X, end_Y
    end_X = random.randint(0, 5)
    end_Y = random.randint(4, 5)

    # Set the end goal point as 4
    maze_size[end_X][end_Y] = 4

    # Create a barrier to the maize
    barrier = 0
    while barrier < 4:
        barrier_X = random.randint(0, 5)
        barrier_Y = random.randint(0, 5)

        # Check if barrier falls on the index values of start
        barrier_point = maze_size[barrier_X][barrier_Y]
        if barrier_point == 2 or barrier_point == 4 or barrie
            barrier = barrier
            continue
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\python.exe "F:/IIT - Artificail Intelligence & Data Science/CM2602 Artificial
--- 6 x 6 Maze ---
-----DFS ALGORITHM-------
0 0 0 0 0 1
0 0 0 1 0 0
0 2 0 0 0 0
0 1 0 0 4 0
0 1 0 0 0 0
0 0 0 0 0 0

-----DFS path found-----
DFS path travelled: [(2, 1), (2, 1), (2, 2), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 4), (4, 5), (5, 4), (5, 5), (5, 3), (4, 3)]
Time taken by DFS to find the goal node: 21 minutes

------a* algorithm------
Heuristic cost:  [4, 3, 3, 3, 3, 3, 4, 3, 2, 2, 2, 2, 4, 3, 2, 1, 1, 1, 4, 3, 2, 1, 0, 1, 4, 3, 2, 1, 1, 1, 4, 3, 2, 2, 2, 2]

a* path travelled:
[(1, 1)][(0, 2)][(2, 1)]
Time taken by a* to reach the goal node: 3 minutes


Process finished with exit code 0
```

Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (21 minutes ago)

File Edit View Navigate Code Refactor Run Tools Git Window Help
main.py - F:\IIT - Artificail Intelligence & Data Science\CM2602 Artificial Intelligence\AI_CW.py
F: IIT - Artificail Intelligence & Data Science CM2602 Artificial Intelligence AI_CW.py
AI_CW.py
Run: AI_CW
ADMIN
ADMIN C:\Users\ADMI
PyCharm Community E
bin
help
jbr
lib
license
plugins
skeletons
build.txt
product-info.json
External Libraries
Scratches and Consoles
create_maze()
Git TODO Problems Terminal Python Packages Python Console Cap Console
Event Log
26:1 CRLF UTF-8 4 spaces Python 3.10 master

Output 3:


File Edit View Navigate Code Refactor Run Tools Git Window Help main.py - F:\IIT - Artificail Intelligence & Data Science\CM2602 Artificial Intelligence\AI_CW.py
IIT - Artificail Intelligence & Data Science 〉 CM2602 Artificial Intelligence 〉 AI_CW.py
AI_CW.py
Run: AI_CW
Externally added files can be added to Git
View Files Always Add Don't Ask Again
Git TODO Problems Terminal Python Packages Python Console Cap Console
Event Log
Externally added files can be added to Git // View Files // Always Add // Don't Ask Again (22 minutes ago)
22:47 CRLF UTF-8 4 spaces Python 3.10 master

```python
import random

print("--- 6 x 6 Maze ---")
# TASK 1
def create_maze():
    # Create a 2D array for the maze
    maze_size = [[0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0]]

    # Set a start goal point between the last two columns
    global start_X, start_Y
    start_X = random.randint(0, 5)
    start_Y = random.randint(0, 1)

    # Setting the start goal point to 2
    maze_size[start_X][start_Y] = 2

    # Set an end goal point between the 1st two columns
    global end_X, end_Y
    end_X = random.randint(0, 5)
    end_Y = random.randint(4, 5)

    # Set the end goal point as 4
    maze_size[end_X][end_Y] = 4

    # Create a barrier to the maize
    barrier = 0
    while barrier < 4:
        barrier_X = random.randint(0, 5)
        barrier_Y = random.randint(0, 5)

        # Check if barrier falls on the index values of start
        barrier_point = maze_size[barrier_X][barrier_Y]
        if barrier_point == 2 or barrier_point == 4 or barrie
            barrier = barrier
            continue
```

```
C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\python.exe "F:/IIT - Artificail Intelligence & Data Science/CM2602 Artificial
--- 6 x 6 Maze ---
-----DFS ALGORITHM-------
0 0 0 0 0 0
0 0 0 0 4 0
0 0 0 0 0 0
0 0 0 0 0 0
2 1 0 0 1 0
0 0 0 1 0 1

-----DFS path found-----
DFS path travelled: [(4, 0), (4, 0), (5, -1), (5, 0), (5, 1), (5, 2), (4, 2), (4, 3), (3, 3), (3, 4), (3, 5), (4, 4), (4, 5), (5, 3),
Time taken by DFS to find the goal node: 28 minutes

------a* algorithm------
Heuristic cost:  [4, 3, 2, 1, 1, 1, 4, 3, 2, 1, 0, 1, 4, 3, 2, 1, 1, 1, 4, 3, 2, 2, 2, 2, 4, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4]

a* path travelled:
[(0, 3)][(1, 3)][(0, 5)]
Time taken by a* to reach the goal node: 3 minutes


Process finished with exit code 0
```

Completeness:

- DFS is said to be incomplete as it won't always be able to find the goal. In the maze game that was generated there were instances where the DFS algorithms couldn't find its goal node. However, A* can be considered as complete since the algorithm guarantees to find the solution everytime (which means it will always find the goal state everytime in the maze game). A* will always use its most potential path to find the goal node.

Optimality:

- Optimality is the ability of a certain algorithm to produce its most optimal solution for a given task. When observing the output of DFS and A* algorithms in the maze game, it clearly shows that A* star is more optimal than the DFS algorithm. The reason for this is, DFS runs in a longer path than A* which takes a long time for the algorithm to find the goal node. A* is optimal as its finds the goal node by moving in the shortest path.

Time complexity:

- The time complexity of DFS in a maze game depends on how the algorithm has been implemented. However the time complexity of DFS will be O(b^m). But the time complexity of A* depends with the heuristic cost. The outputs of the maze game shows that A* takes just a couple of minutes to find its goal node while DFS takes long time to execute since it explores all the nodes until it find its goal node.

Eg:

1. In output 1, time taken by DFS algorithm to execute is 20mins while A* takes only 2mins
2. In output 2, time taken by DFS algorithm to execute is 21mins while A* takes only 3mins
3. In output 3, time taken by DFS algorithm to execute is 28mins while A* takes only 3mins

References:

Virus using discrete event simulator (no date) *NetLogo Models Library: Virus Using Discrete Event Simulator*. Available at: https://ccl.northwestern.edu/netlogo/models/VirusUsingDiscreteEventSimulator (Accessed: December 30, 2022).

Auction algorithms for Network Flow Problems: A tutorial introduction 1 (no date). Available at: https://dspace.mit.edu/bitstream/handle/1721.1/3265/P-2108-26912652.pdf (Accessed: December 30, 2022).

Naeem, M.A. (2021) *A-star (A*) search for solving a maze using python (with visualization)*, *Medium*. Level Up Coding. Available at: https://levelup.gitconnected.com/a-star-a-search-for-solving-a-maze-using-python-with-visualization-b0cae1c3ba92 (Accessed: December 30, 2022).

Blades, A. (2020) *Solving mazes with depth-first search*, *Medium*. The Startup. Available at: https://medium.com/swlh/solving-mazes-with-depth-first-search-e315771317ae (Accessed: December 31, 2022).