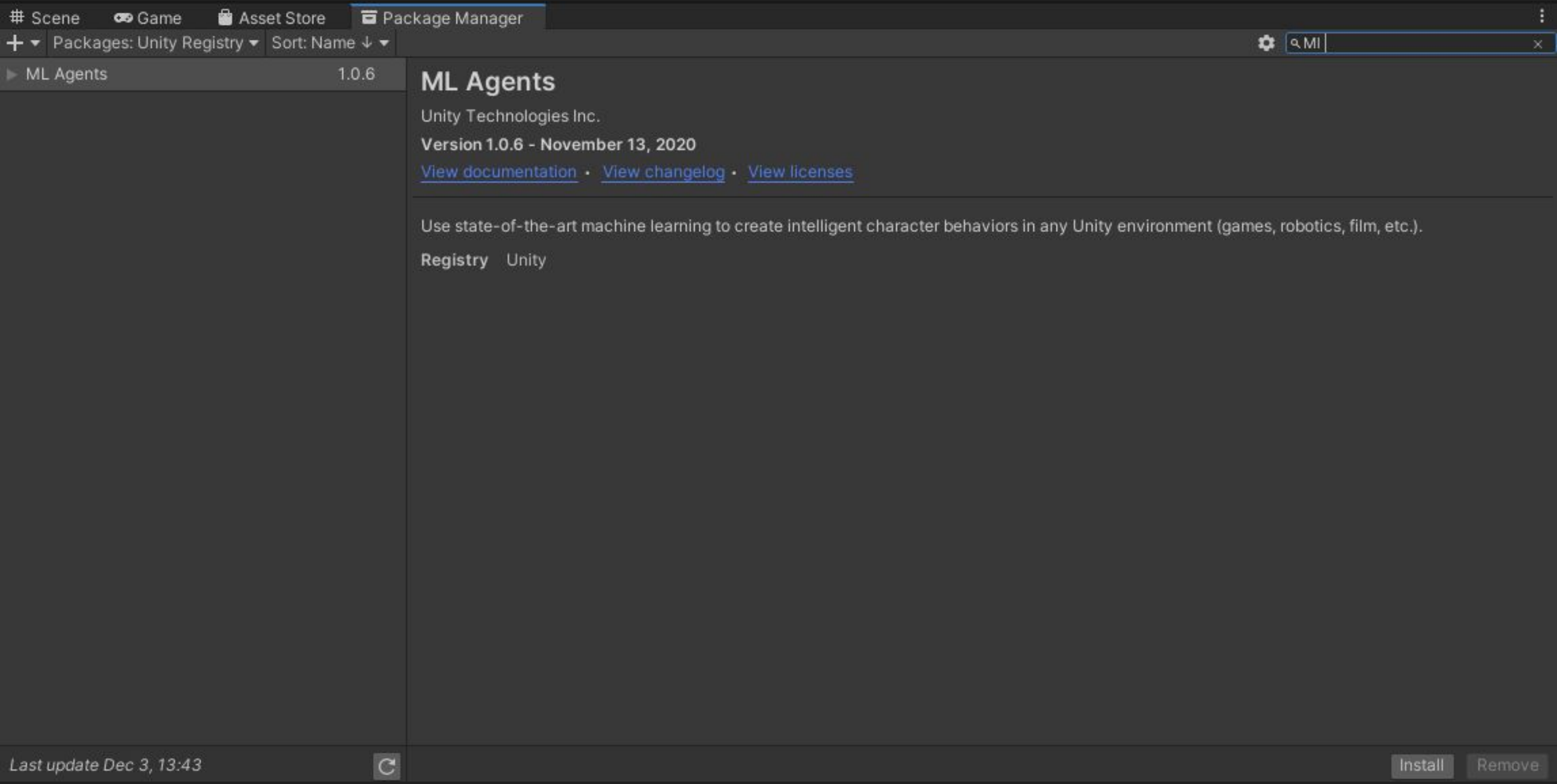
Step 1: Install the MLAgents Package

- Navigate to the Unity package manager
- Change package source to be unity registry
- Search for ML Agents
- Install



ML Agents 1.0.6

ML Agents

Unity Technologies Inc.

Version 1.0.6 - November 13, 2020

View documentation · View changelog · View licenses

Use state-of-the-art machine learning to create intelligent character behaviors in any Unity environment (games, robotics, film, etc.).

Registry Unity

Last update Dec 3, 13:43

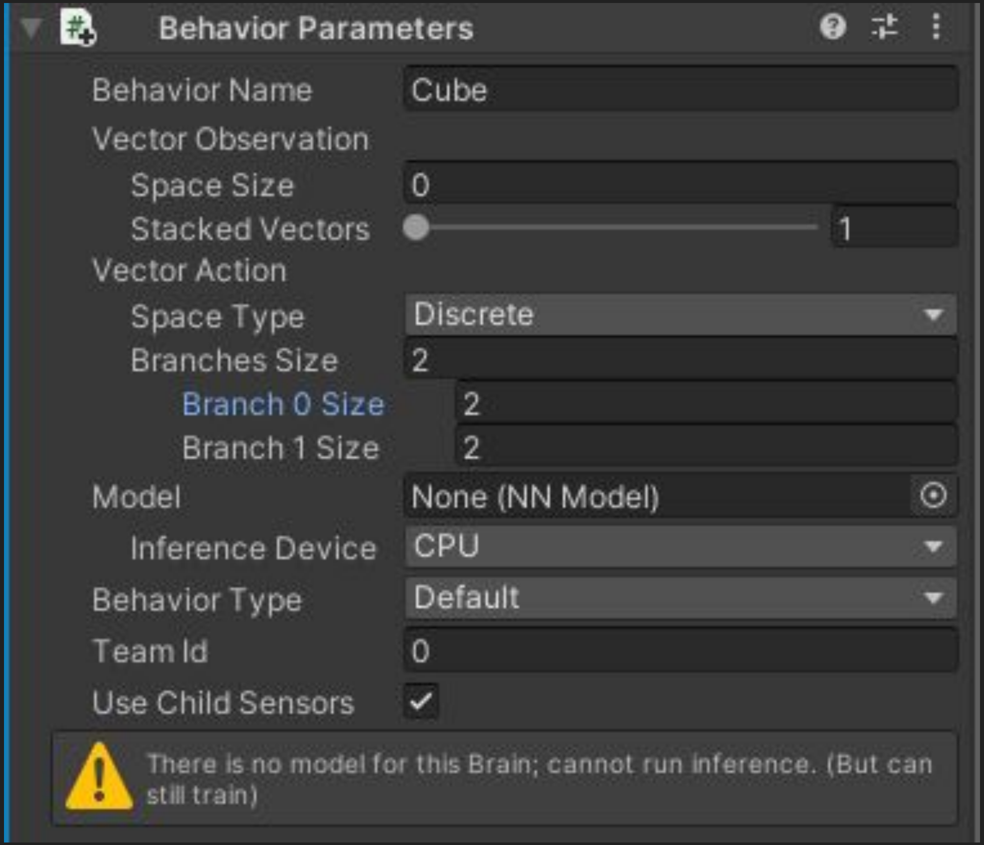
Step 2: Add behavior parameters

Now we need to give our AI the tools it needs to succeed in its environment!

- Under environment, select the cube
    - Scroll to the bottom of the inspector and click Add Component
    - Search for “Behaviour Parameters” and add that component to your cube

Step 3: Behaviour Parameters

- Set name to “Cube”
  - Or whatever you want, we just want to give this guy a name :)
- Set Vector Observation Space Size to 0
  - This is because we will not be manually adding any observations to the vector containing our observations.
- Under Vector Actions, Set the branches size to be 2
  - This is because there are two possible actions our agent can make (left / right)
- Set the size to be 2 for both Branch 0 and Branch 1
  - This is because both left and right can take in two discrete values (true or false)
  - Think of these true or false values as the agent is either currently moving in a direction, or it is not.



Behavior Parameters

Behavior Name	Cube
Vector Observation	
Space Size	0
Stacked Vectors	1
Vector Action	
Space Type	Discrete
Branches Size	2
Branch 0 Size	2
Branch 1 Size	2
Model	None (NN Model)
Inference Device	CPU
Behavior Type	Default
Team Id	0
Use Child Sensors	✓

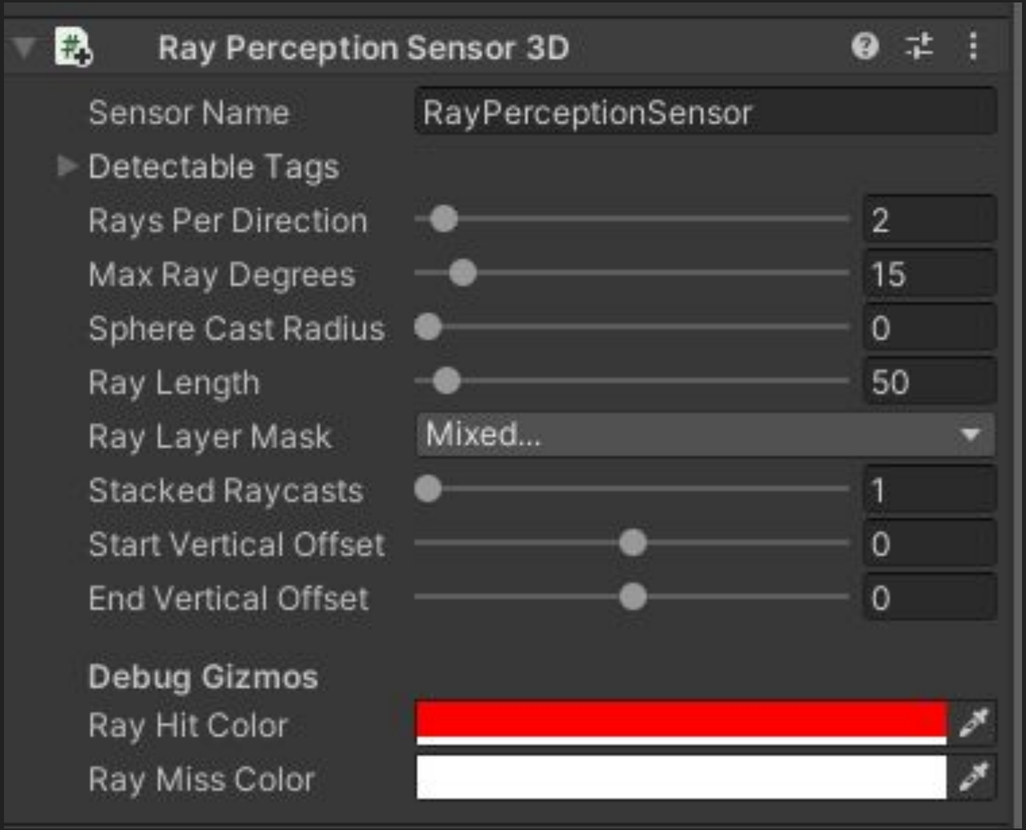
There is no model for this Brain; cannot run inference. (But can still train)

Step 4: Adding Vision

The behavior parameter component has allowed our agent to move left and right, but it currently has no context for the movements it would be making. We need to give our agent a way to see incoming blocks. This is done through the ray perception sensor component.

Ray Perception Sensor:

- Again under environment, select the cube
  - Scroll to the bottom of the inspector and click Add Component
  - Search for “Ray Perception Sensor” and add **Ray Perception Sensor 3D**
- **Set these values:**
  - Rays per direction to 2
  - Max Ray Degrees to 15
  - Sphere cast radius to 0
  - Ray Length to 50
- We won’t be going into detail on what these values specifically are, however, please feel free to play with them and find what works best for you, these values are just what I had the most success with.


Ray Perception Sensor 3D

Sensor Name	RayPerceptionSensor
Detectable Tags	
Rays Per Direction	2
Max Ray Degrees	15
Sphere Cast Radius	0
Ray Length	50
Ray Layer Mask	Mixed...
Stacked Raycasts	1
Start Vertical Offset	0
End Vertical Offset	0

Debug Gizmos

Ray Hit Color	🟥
Ray Miss Color	⬜

Step 5: Connection

We can think of the state of our cube as it having eyes and a brain, but no connection between the two - let’s go ahead and connect those bad boys :)

- Under Assets, navigate to the Scripts folder
- Double click on “CubeScript” to open this script

Imports

First we will need to import a couple libraries into our CubeScript

- At the top, add the line “using System;”
- At the bottom of the import list, add the line “using Unity.MLAgents;”

Variables

Add the following line of code above the start method:

“public event Action OnReset;”

This is used by the MLAgents framework to handle resets when training.

Changes to methods

The first method we are updating is the Start() method.

- Change “void Start()” to **“public override void Initialize()”**
  - Contents in this method remain the same

Next we will update the “FixedUpdate()” method.

- Replace all the code within the method body with a single method call “RequestDecision()”
  - This method call requests a decision from the Agent on what action should be taken, and our movement will be controlled based on that

New Methods:

We will be adding two new methods which are crucial for the movement of the Agent

- public override void OnActionReceived(float[] vectorAction)
- public override void Heuristic(float[] actionsOut)

public override void OnActionReceived(float[] vectorAction)

This method is used to receive actions from the Agents brain, and translate them to movement.

- This is the code which should go in the method body:

```
        //Move left
        if(Mathf.FloorToInt(vectorAction[0])==1)
        {
            Move(1);
        }
        //Move Right
        if(Mathf.FloorToInt(vectorAction[1])==1)
        {
            Move(0);
        }
```

public override void Heuristic(float[] actionsOut)

Another aspect of the agents movement controls, here is where we take in input and decide whether or not we are moving.

- This is the code which should go in our method body:

```
    public override void Heuristic(float[] actionsOut)
  {
      // set moveleft and moveright to be 0
      actionsOut[0] = 0;
      actionsOut[1] = 0;
      if(Input.GetKey(leftKey))
      {
          Move(1);
          //indicate we're movin and groovin
          actionsOut[0] = 1;
      }
      if(Input.GetKey(rightKey))
      {
          Move(0);
          actionsOut[1] = 1;
      }
  }
```

Reset()

Within the reset method there is only one line we need to add

- OnReset?.Invoke();
    - This will allow the MLAgents framework to use our reset method when needed

Method to call at the start of each new session

```
    public override void OnEpisodeBegin()
    {
        Reset();
    }
```

Reinforcement Learning

- Agents learn via trial-and-error
- Two types of feedback
  - Positive Feedback (Reward)
  - Negative Feedback (Punishment)
- Goal of reinforcement learning
  - Find the best sequence of actions that will allow the agent to solve a problem while maximizing a long term reward.

Essentially, the Agent will make an action, and receive feedback on this action to maximize the amount of positive feedback it receives.

Implementing Reinforcement Learning:

- Two more functions we need to update
  - OnCollisionEnter
  - OnTriggerEnter

OnCollisionEnter

We need to replace Reset(); with the following two lines of code:

```
// punish our boi (assign negative reward)
AddReward(increment: -1.0f);
// Reset the training episode
EndEpisode();
```

*Note, if you downloaded the repo and there was a second if statement checking for collision with score, please delete this if statement, it is not needed.

OnTriggerEnter

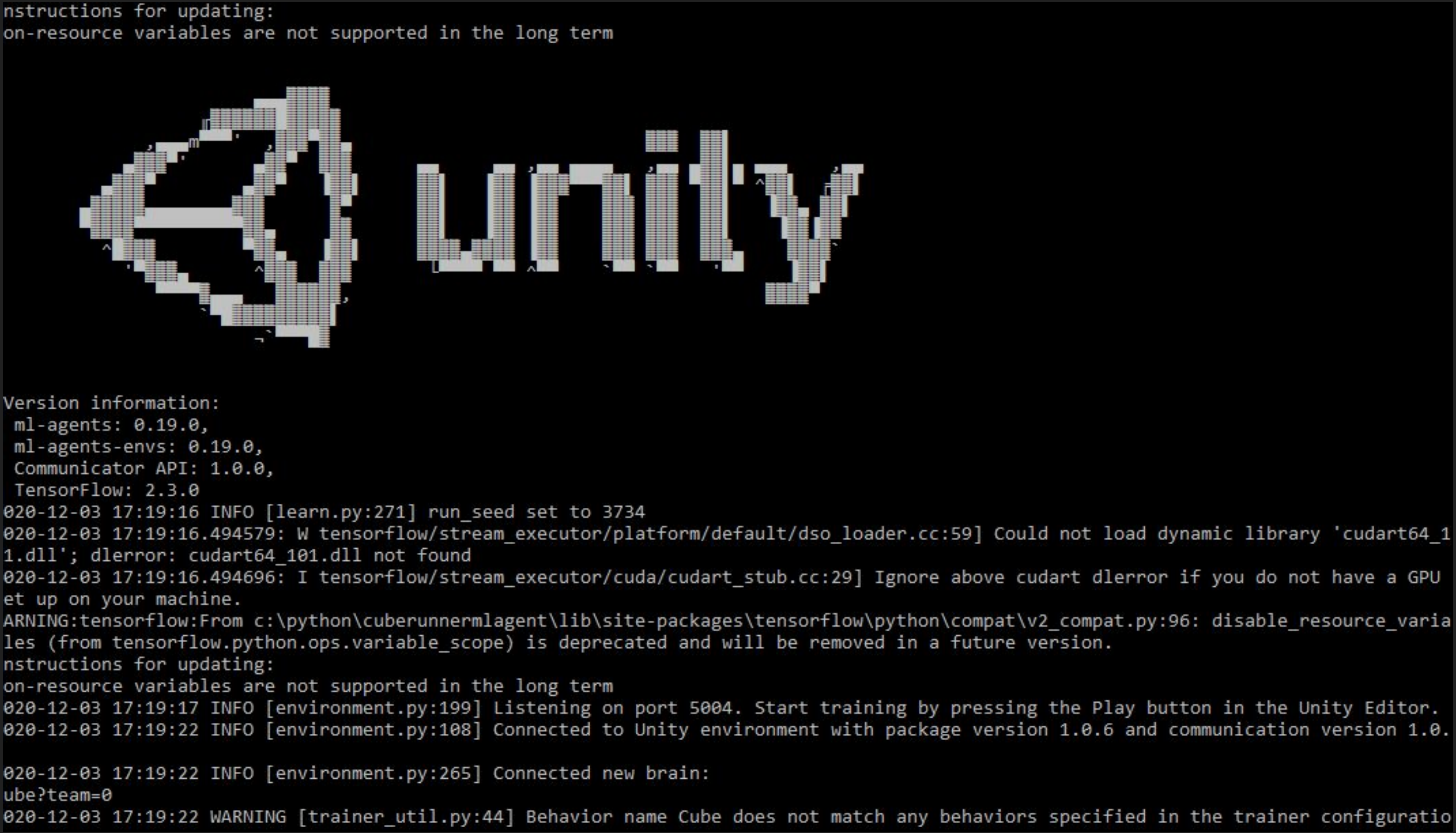
When this method is activated, we are checking to see if we have passed through the gap in the wall. The goal of this game is to pass through as many gaps as possible and as such, we need to assign a positive reward to the Agent

- Add this line to the method: `AddReward(increment: -1.0f);`

What’s next?

At this point, we have updated all the necessary code, and are ready to begin training.

- Open your command prompt and activate the virtual environment that should previously have been set up.
- Once our virtual environment has been activated, we want to type the following command into our command line
  - mlagents-learn --force
    - Force is not necessary, if we have previously trained a model this will just override it



```
nstructions for updating:
on-resource variables are not supported in the long term
```

```
Version information:
 ml-agents: 0.19.0,
 ml-agents-envs: 0.19.0,
 Communicator API: 1.0.0,
 TensorFlow: 2.3.0
020-12-03 17:19:16 INFO [learn.py:271] run_seed set to 3734
020-12-03 17:19:16.494579: W tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load dynamic library 'cudart64_1
1.dll'; dlerror: cudart64_101.dll not found
020-12-03 17:19:16.494696: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU
et up on your machine.
ARNING:tensorflow:From c:\python\cuberunnermlagent\lib\site-packages\tensorflow\python\compat\v2_compat.py:96: disable_resource_varia
les (from tensorflow.python.ops.variable_scope) is deprecated and will be removed in a future version.
nstructions for updating:
on-resource variables are not supported in the long term
020-12-03 17:19:17 INFO [environment.py:199] Listening on port 5004. Start training by pressing the Play button in the Unity Editor.
020-12-03 17:19:22 INFO [environment.py:108] Connected to Unity environment with package version 1.0.6 and communication version 1.0.

020-12-03 17:19:22 INFO [environment.py:265] Connected new brain:
ube?team=0
020-12-03 17:19:22 WARNING [trainer_util.py:44] Behavior name Cube does not match any behaviors specified in the trainer configuratio
```

Now what?

Click the play button in unity to begin training :)