What is Math Parser
The algorithm of Math Parser, is to tokenize the math expression into individual meaningful symbols, and apply shunting-yard algorithm for output.

Supported Operation
Sin,cos, tan, arcsine, arccos, atan, sqrt, abs, floor, ceil, round, min, max, rand, atan2, pow, clamp, sign, lerp, smoothstep, step

Initialization

```
Using MathExpParser;

MathParser marseParser = new MathParser();
```

Execution

```
string sampleExpression = "90 + pow(5, 2)";
//Answer = 115
float syncAnswer = mathParser.Calculate(sampleExpression);
```

Async/Await solution

```
string sampleExpression = "90 + pow(5, 2)";
mathParser.CalculateAsyn(sampleExpression , (float answer)
{
    Debug.Log("Async Answer " + answer);
});
```

Threading

You may choose to add MathParserThreading on your scene, or else it will add automatically.
Threading method provides much more power horse than the rest solution. However, couldn't work on WebGL before Unity 2019

```
MathParserThreading.Instance.CalculateAsyn("5+50*(1.2)",
(MathParserThreading.ParseResult result) =>
      {
           Debug.Log("Answer " + result.answer);
      });
```

Customize math symbol

```
Dictionary<string, float> testDict = new Dictionary<string, float>
{
    {"p", 5 },
    { "t", 3}
};

string sampleExpression = "340 + (sin(t) * p)";
//Answer = 340.7056
float syncAnswer = mathParser.Calculate(sampleExpression);
```

Add Math operate functions

1. Open ShuntingYardParser.cs
2. Insert you own function name into Dictionary FunctionLookUpTable,
3. Key = function name, variable = number of parameters needed
4. Finally add the actual operation into Function ComputeFunctionToken

Enable Cache

```
//Default true
//Once enabled, it will remember all your output given by math
//expression. Hugely boost performance.
mathParser.cacheMode = true;
```

Operation definition

```
sin(float x)
cos(float x)
tan(float x)
arcsine(float x)
arccos(float x)
atan(float x)
sqrt(float x)
abs(float x)
floor(float x)
ceil(float x)
round(float x)
sign(float x)

step(float edge, float x)
min(float x, float y)
max(float x, float y)
```

```
atan2(float y, float x)
pow(float value, float power)

clamp(float value, float min, float max)
lerp(float start, float end, float t)
smoothstep(float start, float end, float t)
```