In [1]:
```
%cd '/content/drive/MyDrive/project2'
```
```
/content/drive/MyDrive/project2
```

In [2]:
```
import numpy as np
import pandas as pd
from supervisor import *
```

MCTS Experiment

I will play ordinary and improve 200 times (100 for improve vs ordinary, 100 for ordinary vs improved) for each of the given times below:

1. t=.001
2. t=.01
3. t=.03
4. t=.05*

*50x

Finally, I'll play 10 games (5 for improve vs ordinary, 5 for ordinary vs improve for t=.25) (for the sanity of my RAM)

I've modified the supervisor code such that it returns a tuple/doesn't print anything out:

vals[0]: player 1 win (0:no, 1:yes)

vals[1]: player 2 win (0:no, 1:yes)

In [5]:
```
ordVImprovet1, ordVImprovet2, ordVImprovet3, ordVImprovet4 = [],[],[],[]
improveVOrd1, improveVOrd2, improveVOrd3, improveVOrd4 = [],[],[],[]
```

Grabbing Stats:

`getting stats at t = .001:`

In [5]:
```
for i in range(0, 100):
  val = supervisor("ordinary", "improved", 0.001, 0)
  ordVImprovet1.append(val)
  val1 = supervisor("improved", "ordinary", 0.001, 0)
  improveVOrd1.append(val1)
```

In [15]:
```
improveWins0 = sum(map(lambda x: x[0], improveVOrd1))
print("when the improved alg is p1:", improveWins0,"of the 100 games are w
on")

improveWins = sum(map(lambda x: x[1], ordVImprovet1))
print("when the improved alg is p2:", improveWins,"of the 100 games are wo
n")
```
```
when the improved alg is p1: 75 of the 100 games are won
when the improved alg is p2: 82 of the 100 games are won
```

`getting stats at t = .01:`

In [16]:
```
for i in range(0, 100):
  val = supervisor("ordinary", "improved", 0.01, 0)
  ordVImprovet2.append(val)

  val1 = supervisor("improved", "ordinary", 0.01, 0)
  improveVOrd2.append(val1)
```

In [18]:
```
improveWins0 = sum(map(lambda x: x[0], improveVOrd2))
print("when the improved alg is p1:", improveWins0,"of the 100 games are w
on")

improveWins = sum(map(lambda x: x[1], ordVImprovet2))
print("when the improved alg is p2:", improveWins,"of the 100 games are wo
n")
```
```
when the improved alg is p1: 84 of the 100 games are won
when the improved alg is p2: 79 of the 100 games are won
```

`getting stats at t = .03:`

In [5]:
```
for i in range(0, 100):
  val = supervisor("ordinary", "improved", 0.03, 0)
  ordVImprovet3.append(val)

  val1 = supervisor("improved", "ordinary", 0.03, 0)
  improveVOrd3.append(val1)
```

In [6]:
```
improveWins0 = sum(map(lambda x: x[0], improveVOrd3))
print("when the improved alg is p1:", improveWins0,"of the 100 games are w
on")

improveWins = sum(map(lambda x: x[1], ordVImprovet3))
print("when the improved alg is p2:", improveWins,"of the 100 games are wo
n")
```
```
when the improved alg is p1: 85 of the 100 games are won
when the improved alg is p2: 70 of the 100 games are won
```

`getting stats at t = .05:`

In [6]:
```
for i in range(0, 50):
  val = supervisor("ordinary", "improved", 0.05, 0)
  ordVImprovet4.append(val)

  val1 = supervisor("improved", "ordinary", 0.05, 0)
  improveVOrd4.append(val1)
```

In [8]:
```
improveWins0 = sum(map(lambda x: x[0], improveVOrd4))
print("when the improved alg is p1:", improveWins0,"of the 50 games are wo
n")

improveWins = sum(map(lambda x: x[1], ordVImprovet4))
print("when the improved alg is p2:", improveWins,"of the 50 games are won
")
```
```
when the improved alg is p1: 42 of the 50 games are won
when the improved alg is p2: 37 of the 50 games are won
```

`getting stats at t = .25:`

In [3]:
```
ordVImprovet5, improveVOrd5 = [],[]
```

In [4]:
```
for i in range(0, 10):
  val = supervisor("ordinary", "improved", 0.25, 0)
  ordVImprovet5.append(val)

  val1 = supervisor("improved", "ordinary", 0.25, 0)
  improveVOrd5.append(val1)
```

In [5]:
```
improveWins0 = sum(map(lambda x: x[0], improveVOrd5))
print("when the improved alg is p1:", improveWins0,"of the 10 games are wo
n")

improveWins = sum(map(lambda x: x[1], ordVImprovet5))
print("when the improved alg is p2:", improveWins,"of the 10 games are won
")
```
```
when the improved alg is p1: 9 of the 10 games are won
when the improved alg is p2: 2 of the 10 games are won
```

In [8]:
```
ordVImprovet5
```
Out[8]:
```
[(1, 0, 6),
 (1, 0, 30),
 (0, 1, 1),
 (1, 0, 2),
 (0, 1, 8),
 (1, 0, 20),
 (1, 0, 36),
 (1, 0, 6),
 (0, 0, 0),
 (0, 0, 0)]
```

In this experiment, I've played improved vs ordinary multiple times with a variety of times associated. In most cases, improved.py tends to perform significantly better. While only 2 games were won by improved in t=.25, the sample size is incredibly small and 2 of the games are tied. In addition, player 1 does have the tendency to have a strategic advantage.