





















שאלה 1:

Bot	רצף קבוע	מטה- אסטרטגיה	רנדומי
Antiflat			
Bruijn81			
Copy			
flat			
foxtrot			
Freq			
pi			
Play226			
rndplayer			
rotate			
switch			
switchalot			
uzra			
megahal			
iocaine			
greenberg			

שאלה 2:

סטוכסטי

שאלה 3:

הסוכן שלנו מנסה לאמוד את התפלגות
המהלכים שלו

שאלה 4:

עושה את שניהם הוא עושה למשל
exploitation בשלב ה mutualism ולמשל
עושה exploration בשלב elitism

שאלה 5+6:

כמו במעבדה הקודמת היה לנו host שהוא
האוכלוסייה שלנו האתחול שלנו היה אתחול
רנדומאלי בצורה הזאת

```
def init_population(self):  
    for i in range(self.popsiize):  
        Citizen = Agent()  
        self.population.append(Citizen)  
        self.buffer.append(Citizen)
```

```

def __init__(self):
    super().__init__()
    self.moves = []
    self.trial = 1000
    self.score=0
    self.fitness=0
    for i in range(1000):
        self.moves.append(randint(0,2))

```

ואז גם עשינו parasites שהם היריבים
אתחלנו

```

def addPlayers(self):
    self.participants.append(AntiFlat())
    self.participants.append(Copy())
    self.participants.append(Freq())
    self.participants.append(Flat())
    self.participants.append(Foxtrot())
    self.participants.append(Bruijn81())
    self.participants.append(Pi())
    self.participants.append(Play226())
    self.participants.append(RndPlayer())
    self.participants.append(Rotate())
    self.participants.append(Switch())
    self.participants.append(SwitchALot())
    self.participants.append(Iocaine())
    self.participants.append(Greenberg())
    self.participants.append(Urza())
    self.participants.append(MegaHal())
    for _ in range(len(self.participants)):
        self.score.append(0)
        self.finalScore.append(0)

```

ואז הפיטנס היה כמה נקודות השחקן הזה
קיבל

ואז כדי לעשות mutate השתמשנו ב 3
השלבים שראינו בהרצאה

Mutualism phase:

```
def mutualism_phase(self, best):
    for i in range(self.popsiz):
        index = randint(0, self.popsiz - 1)
        while index == i:
            index = randint(0, self.popsiz - 1)
        BF1 = randint(1, 2)
        BF2 = randint(1, 2)
        mutual_vector = []
        for j in range(len(self.population[i].moves)):
            num = int((self.population[i].moves[j] + self.population[index].moves[j]) / 2)
            mutual_vector.append(num)
        newvec1, newvec2 = [], []

        for j in range(len(self.population[i].moves)):
            num1 = abs(int(self.population[i].moves[j] + uniform(0, 1) * (best.moves[j] - mutual_vector[j] * BF1)))
            num2 = abs(
                int(self.population[index].moves[j] + uniform(0, 1) * (best.moves[j] - mutual_vector[j] * BF2)))
            newvec1.append(num1 % 3)
            newvec2.append(num2 % 3)

        finalscore = 0
        my_agent = Agent()
        my_agent.moves = newvec1
        for player in range(len(self.participants)):
            finalscore += self.startGame2(my_agent, player)
        if finalscore > self.population[i].fitness:
            self.population[i].moves = newvec1
            self.population[i].fitness = finalscore
```

```
finalscore = 0
my_agent2 = Agent()
my_agent2.moves = newvec2
for player in range(len(self.participants)):
    finalscore += self.startGame2(my_agent2, player)
if finalscore > self.population[index].fitness:
    self.population[i].moves = newvec1
    self.population[i].fitness = finalscore
```

אחרי שמחשבים גנים חדשים לפי הנוסחה בודקים אם הוא יותר טוב ממה שהיה ואז מחליפים

Commensalism phase:

```
def Commensalism_phase(self, best):
    for i in range(self.popsiz):
        index = randint(0, self.popsiz - 1)
        while index == i:
            index = randint(0, self.popsiz - 1)
        newvec = []
        for j in range(len(self.population[i].moves)):
            num = abs(int(self.population[i].moves[j] + random.choice([-1, 1]) * (
                best.moves[j] - self.population[index].moves[j])))
            newvec.append(num % 3)
        my_agent = Agent()
        my_agent.moves = newvec
        finalscore = 0
        for player in range(len(self.participants)):
            finalscore += self.startGame2(my_agent, player)
        if finalscore > self.population[i].fitness:
            self.population[i].moves = newvec
            self.population[i].fitness = finalscore
```

Paratisim phase:

```
def paratisim_phase(self):
    for i in range(self.popsiz):
        index = randint(0, self.popsiz - 1)
        while index == i:
            index = randint(0, self.popsiz - 1)
        sampled_list = random.sample(self.myarr, 50)
        my_agent = Agent()
        my_agent.moves = self.population[index].moves
        for num in sampled_list:
            my_agent.moves[num] = randint(0, 2)
        finalscore = 0
        for player in range(len(self.participants)):
            finalscore += self.startGame2(my_agent, player)
        if finalscore > self.population[i].fitness:
            self.population[i].moves = my_agent.moves
            self.population[i].fitness = finalscore
```

ואז ממשנו את הטורניר שלנו

```
def start(self):
    for player1 in range(len(self.participants) - 1):
        player1History = []
        for player2 in range(player1 + 1, len(self.participants)):
            finalScore = self.startGame(player1, player2)
            if finalScore == 1:
                self.finalScore[player1] += 1
                self.finalScore[player2] -= 1
            else:
                self.finalScore[player2] += 1
                self.finalScore[player1] -= 1

            player1History.append(finalScore)
        self.matchHistory.append(player1History)
```

```
def startGame(self, player1, player2):
    self.participants[player1].newGame(1000)
    self.participants[player2].newGame(1000)
    score1, score2 = 0, 0
    for _ in range(1000):
        move1 = self.participants[player1].nextMove()
        move2 = self.participants[player2].nextMove()
        if (move1 == 0 and move2 == 1) or (move1 == 1 and move2 == 2) or (move1 == 2 and move2 == 0):
            score2 += 1
            score1 -= 1
            self.participants[player2].storeMove(move2, 1)
            self.participants[player1].storeMove(move1, -1)

        elif (move2 == 0 and move1 == 1) or (move2 == 1 and move1 == 2) or (move2 == 2 and move1 == 0):
            score1 += 1
            score2 -= 1
            self.participants[player1].storeMove(move1, 1)
            self.participants[player2].storeMove(move2, -1)
        else:
            self.participants[player1].storeMove(move1, 0)
            self.participants[player2].storeMove(move2, 0)

    self.score[player1] += score1
    self.score[player2] += score2

    if score2 > score1:
        return -1

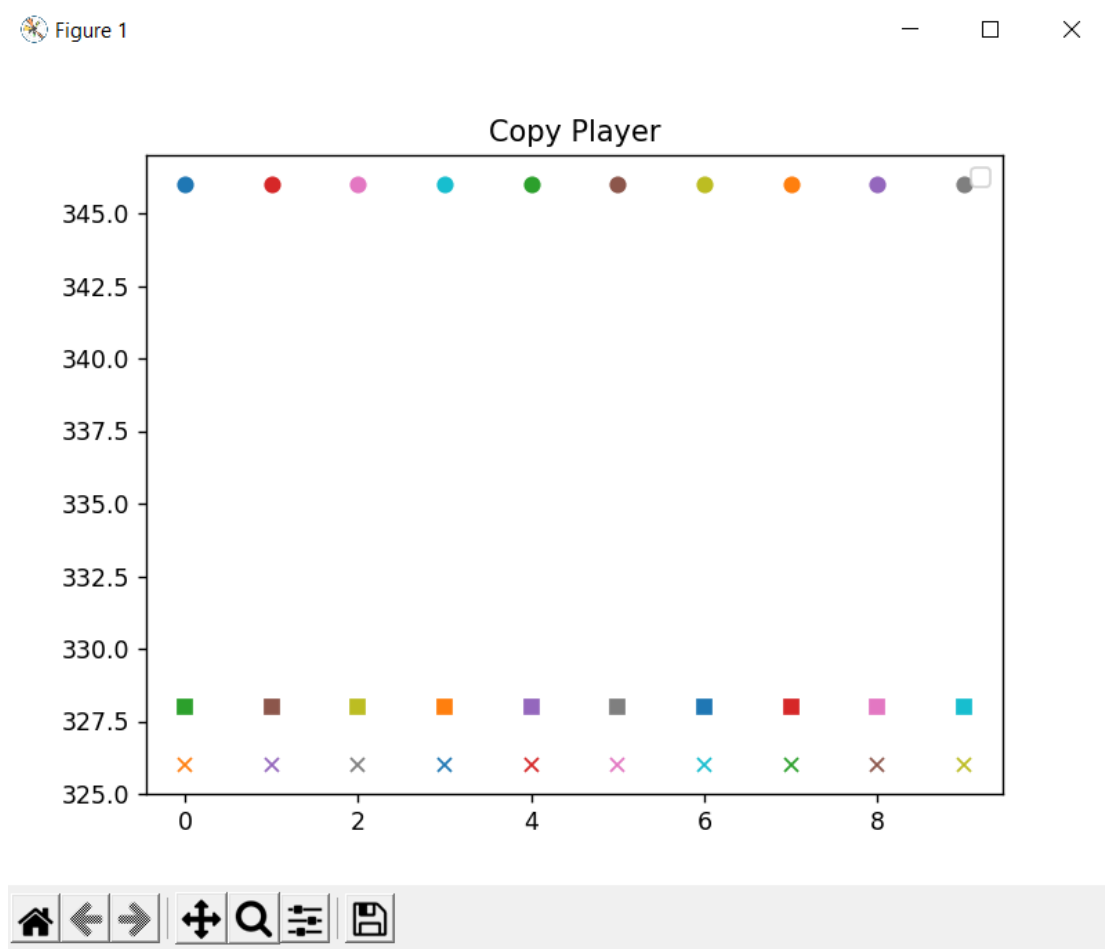
    return 1
```

```
def subOrderPhase(self, best):
```

תוצאות: (דוגמא לחלק מהן)

אז עשינו 10 משחקים נגד כל bot ואז עשינו
שרתות לכל פעם כמה משחקים הוא זכה מ
1000

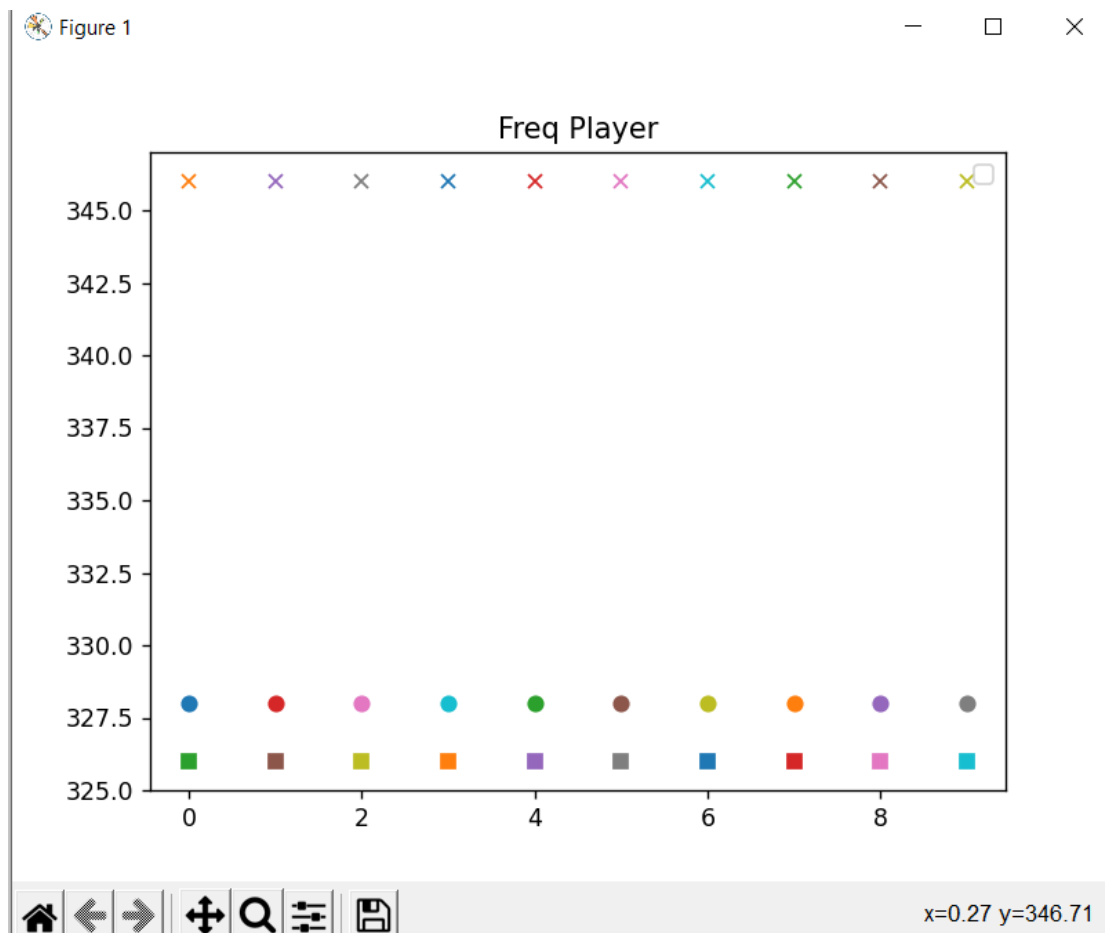
למשל נגד copy player



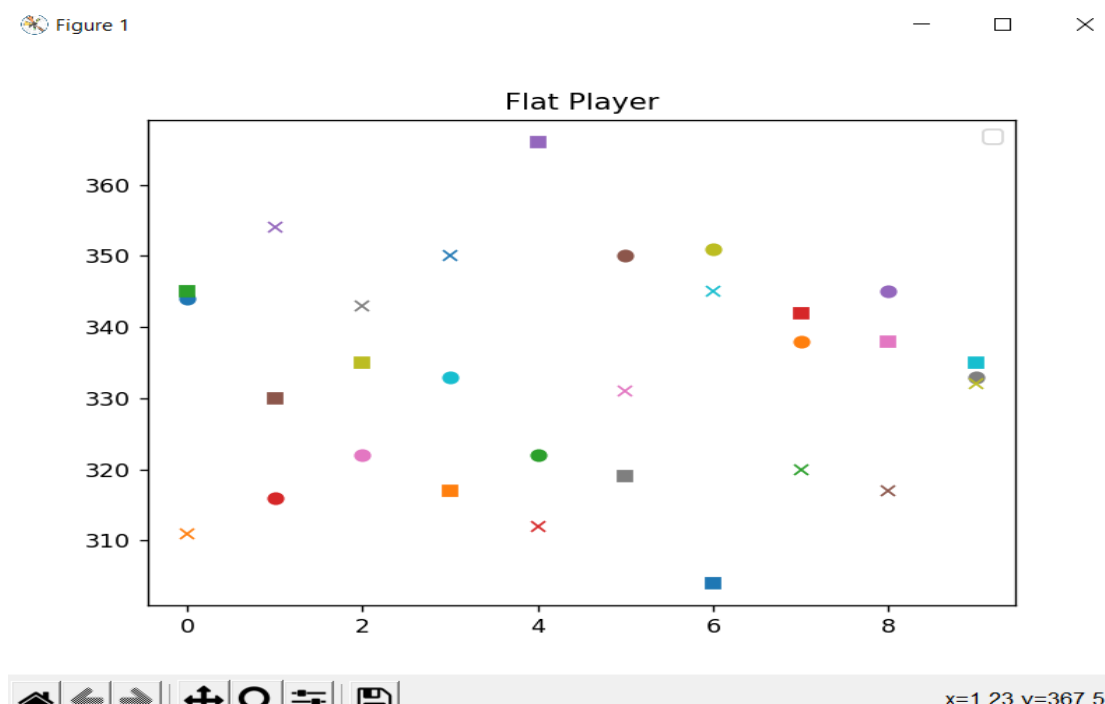
כאשר עיגולים הוא כמה זכה ריבועים זה הפסד
ואיקסים זה תיקו ואז הדפסנו את ממוצע וסטיית
תקן

```
Opp : Copy Player || Win Avg : 346.0 || Win Stdev : 0.0
```

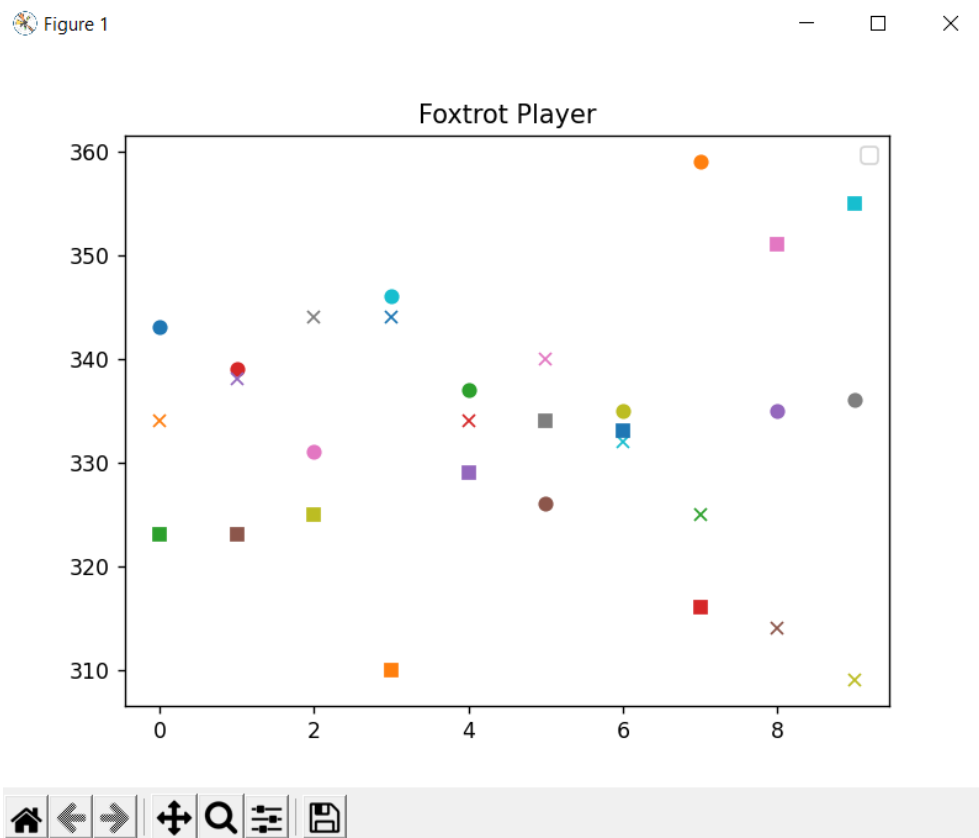
ציור עבור FREQ PLAYER



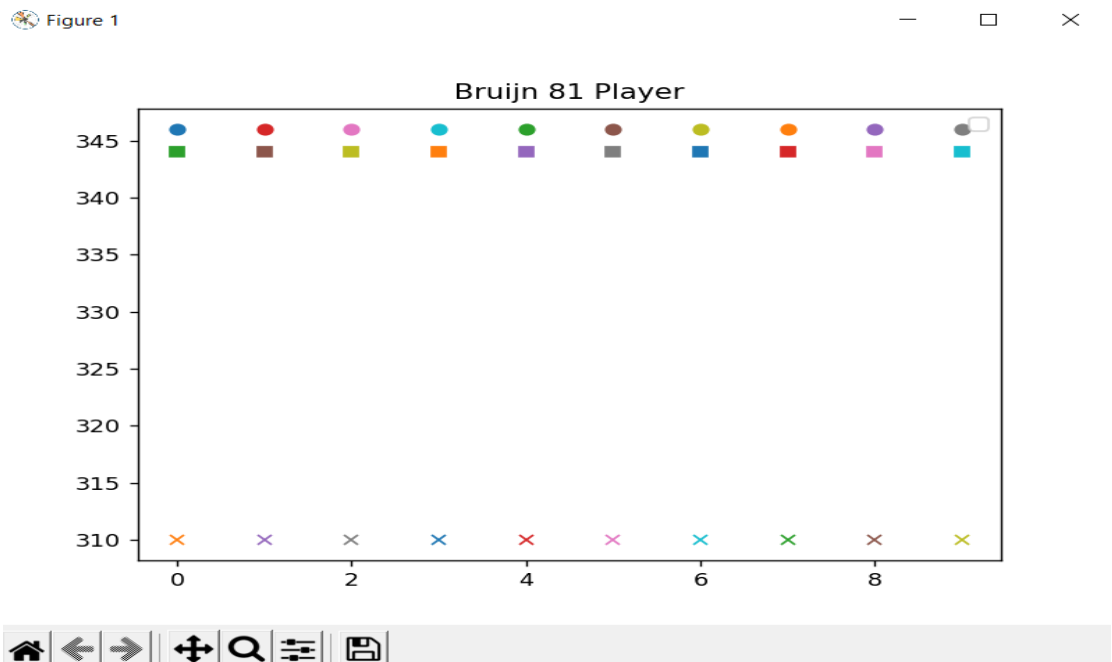
עבור FLAT



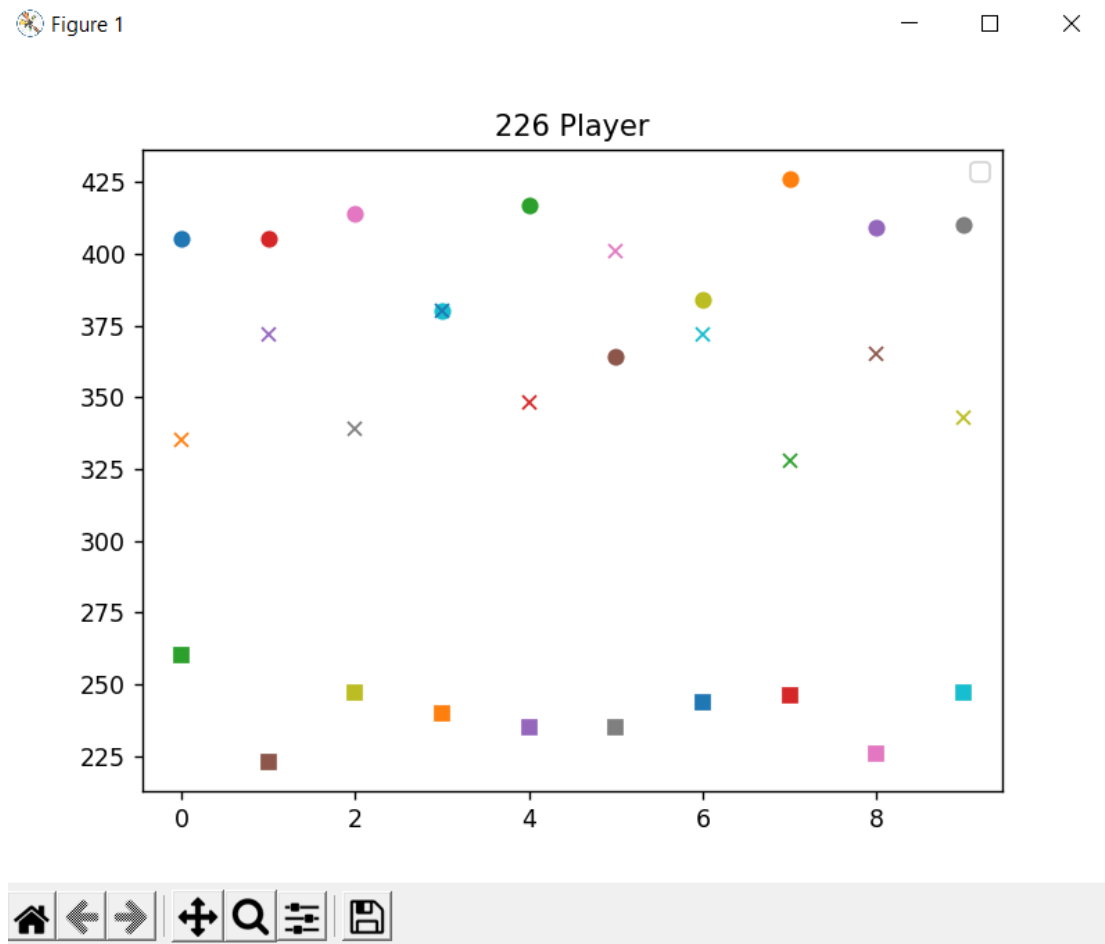
עבור froxtrox



Bruijin81



player 226



עשינו את זה עבור כל bot

```
Opp : Copy Player || Win Avg : 348.0 || Win Stdev : 0.0
Opp : Freq Player || Win Avg : 315.0 || Win Stdev : 0.0
Opp : Flat Player || Win Avg : 337.8 || Win Stdev : 12.976902558006667
Opp : Foxtrot Player || Win Avg : 321.7 || Win Stdev : 15.790644206125489
Opp : Bruijn 81 Player || Win Avg : 334.0 || Win Stdev : 0.0
Opp : Pi Player || Win Avg : 338.0 || Win Stdev : 0.0
Opp : 226 Player || Win Avg : 396.7 || Win Stdev : 15.542415084750074
Opp : Random Player || Win Avg : 337.9 || Win Stdev : 16.339794640352395
Opp : Rotating Player || Win Avg : 348.0 || Win Stdev : 0.0
Opp : Switching Player || Win Avg : 331.7 || Win Stdev : 11.343622780125306
```

ואז דוגמא לסיום הטורניר כשעשינו טורניר מלא

```
My-Agent Score : 12
Switch a Lot Player Score : 5
Pi Player Score : 3
Switching Player Score : 3
Freq Player Score : 1
226 Player Score : 1
Random Player Score : 1
Anti Flat Player Score : -1
Copy Player Score : -1
Flat Player Score : -1
Bruijn 81 Player Score : -1
Rotating Player Score : -3
Foxtrap Player Score : -7
```

סעיף התמודדות עם הבעיות:

כדי להתמודד עם הבעיות מה שעשינו זה היה בחירת גודל אוכלוסייה גדול ואז כאשר עושים mutualism יש יותר וריאנטים מונע יצירת מעגלים דבר אחר שעשינו היה לעשות הגבלה על מספר המופעים של כל BOT ב parasite כדי שהסוכן ילמד נגד כל היריבים ולא רק סוג אחד