THE ITERATED PRISONERS DILEMMA ALLOWS THE STUDY OF COOPERATIVE BEHAVIOUR

- both sides are better off choosing **Cooperation** (3)
- than choosing to **Defect** (1) even so,
- an individual has a **Tempetation** to deviate (5).

	C	D
C	3, 3	5, 0
D	0, 5	1, 1

AXELROD LIBRARY IS AN OPEN SOURCE TOOL THAT COMBINES PYTHON + PRISONERS DILEMMA

- more than 200 condtibutors
- 100% test coverage
- unit and integration tests
- documentation.

WHEN INTERACTING WITH A SNEAKY OPPONENT

	1	2	3	4	5	6
SneakyTitForTat	C	C	D	D	C	C
Grudger	C	C	C	D	D	D

SHOULD PEOPLE HOLD A GRUDGE AGAINST THEM?

```
import axelrod as axl

first_match = axl.Match([axl.SneakyTitForTat(),
                         axl.Grudger()],
                        turns=100)
first_match.play()[:6]
[('C', 'C'), ('C', 'C'), ('D', 'C'),
 ('D', 'D'), ('C', 'D'), ('C', 'D')]

print(first_match.sparklines())

first_match.final_score()
(295, 60)

second_match = axl.Match([axl.TitForTat(),
                          axl.SneakyTitForTat()],
                         turns=100)

second_match.play()
second_match.final_score()
(297, 297)
```

MORE INFORMANTION

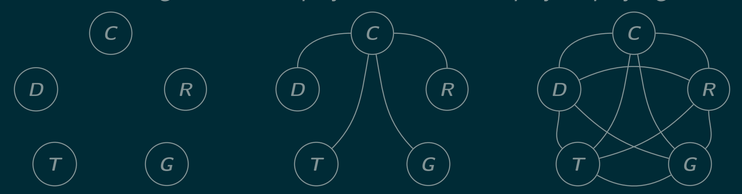
- In case you missed me:
- Github: https://github.com/Axelrod-Python

ABOUT ME

NikoletaGlyn Nikoleta-v3

WHEN FACED WITH DIFEERENT WAR SCENARIOS WHAT IS THE OPTIMAL PLAY?

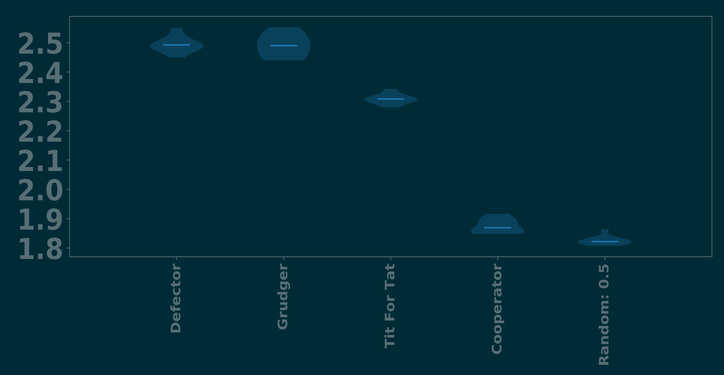
choose strategies *a player interacts* *players play against all*



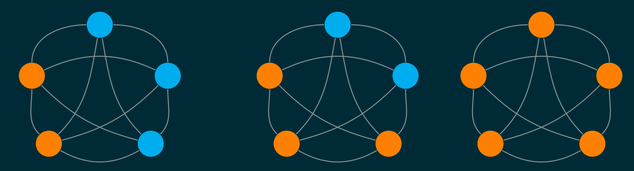

```
import axelrod as axl

axl.seed(0)
players = [axl.Cooperator(), axl.Defector(),
           axl.TitForTat(), axl.Grudger(),
           axl.Random()]
tournament = axl.Tournament(players)
results = tournament.play()
results.ranked_names
['Defector', 'Grudger', 'Tit For Tat',
 'Cooperator', 'Random: 0.5']

plot = axl.Plot(results)
p = plot.boxplot()
p.show()
```



SHOULD THE NORTH JOIN HANDS WITH THE SOUTH TO DEFEAT THE NIGHT KING?



```
import random

N = 5
players = []
axl.seed(5)
for _ in range(N):
    player = random.choice([axl.Defector, axl.Cooperator])
    players.append(player())

mp = axl.MoranProcess(players=players, turns=200)
mp.play()

[Counter({'Cooperator': 3, 'Defector': 2}),
 Counter({'Cooperator': 3, 'Defector': 2}),
 Counter({'Cooperator': 3, 'Defector': 2}),
 Counter({'Cooperator': 2, 'Defector': 3}),
 Counter({'Cooperator': 2, 'Defector': 3}),
 Counter({'Cooperator': 1, 'Defector': 4}),
 Counter({'Cooperator': 1, 'Defector': 4}),
 Counter({'Cooperator': 1, 'Defector': 4}),
 Counter({'Defector': 5})]
```