Axelrod
exploring the iterated prisoner’s dilemma
Axelrod-Python

- **https://github.com/Axelrod-Python**
  - Explore strategies for the Prisoners dilemma game
  - Over 100 strategies from literature and original ones
  - Run round robin tournaments with options
  - Population dynamics (i.e., evolution)

- Easy to install
  - `pip install axelrod`

- Also includes notebooks

- Documentation
Axelrod
A research tool for the Iterated Prisoner's Dilemma

Axelrod Examples
Analysis and examples for the Axelrod-Python library

Axelrod-fingerprint
A repository of fingerprints of all strategies in the Axelrod-Python library

Axelrod-dojo
Trains machine learning strategies for the IPD with evolutionary and particle swarm algorithms, including neural networks and finite state machines

Axelrod-notebooks
A repository of example Jupyter notebooks

tournament
A repository to run the whole suite of strategies in the Axelrod library

Top languages
- Python
- Jupyter Notebook
- Fortran
- TeX
- HTML

People
23
Axelrod Players

• A player like TitForTat is a subclass of a Player class
• Every player subclass has a set of fixed properties (e.g., how many interactions it remembers)
• A subclass has instances with unique IDs
• Instances interact with “opponents”, who are instances of a player subtype
• Each instance maintains a history of its interactions with each opponent it encounters
• Its strategy for an encounter may depend on this
class TitForTat(Player):
    name = "Tit For Tat"
    classifier = {
        "memory_depth": 1,
        "stochastic": False,
        "inspects_source": False,
        "manipulates_source": False,
        ...
    }

def strategy(self, opponent: Player) -> Action:
    # First move
    if not self.history:
        return C
    # React to the opponent's last move
    if opponent.history[-1] == D:
        return D
    return C
class TitFor2Tats(Player):
   """"""player starts by cooperating and then defects only after 2 defects by opponent"""
   name = "Tit For 2 Tats"
   classifier = {
      "memory_depth": 2,
      "stochastic": False,
      "inspects_source": False,
      "manipulates_source": False,
      ...
   }

   @staticmethod
   def strategy(opponent: Player) -> Action:
      return D if opponent.history[-2:] == [D, D] else C
class TitFor2Tats(Player):
    """ player that behaves opposite to Tit For Tat, including first move"""
    name = "Tit For 2 Tats"
    classifier = {
        "memory_depth": 2,
        "stochastic": False,
        "inspects_source": False,
        "manipulates_source": False,
        ...
    }

    @staticmethod
    def strategy(opponent: Player) -> Action:
        return C if opponent.history[-1:] == [D] else D
Predefined Player Strategies

• There are 24 variations on the basic *Tit For Tat* strategy
• And more than 100 other player strategies
• See an index [here](#) with brief descriptions and links to the Python source code