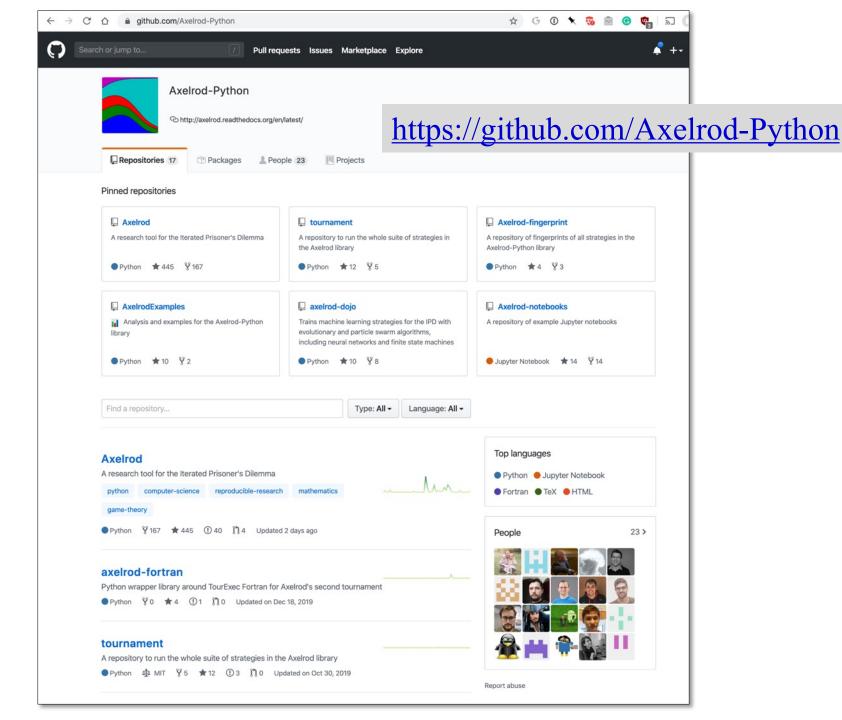
Axelrod

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 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

TitForTat

Remembers only last interaction with a given player

Note use of type hints, added in 3.5

class TitFor2Tats(Player):

TitFor2Tats

```
"""player starts by cooperating and then defects only after 2defects by opponent"""
name = "Tit For 2 Tats"
classifier = {
                                                     Remembers last2
 "memory_depth": 2,-
                                                     interactions with a
 "stochastic": False,
                                                     given player
 "inspects_source": False,
 "manipulates source": False,
  ...}
                                                        Cooperates unless
                                                        this opponent
@staticmethod
                                                       defected last two
def strategy(opponent: Player) -> Action:
                                                        times
  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

Bulley

Predefined Player Strategies

- There are 24 variations on the basic *Tit For Tat* strategy
- And more than 100 other player strategies
- See an index <u>here</u> with brief descriptions and links to the Python source code