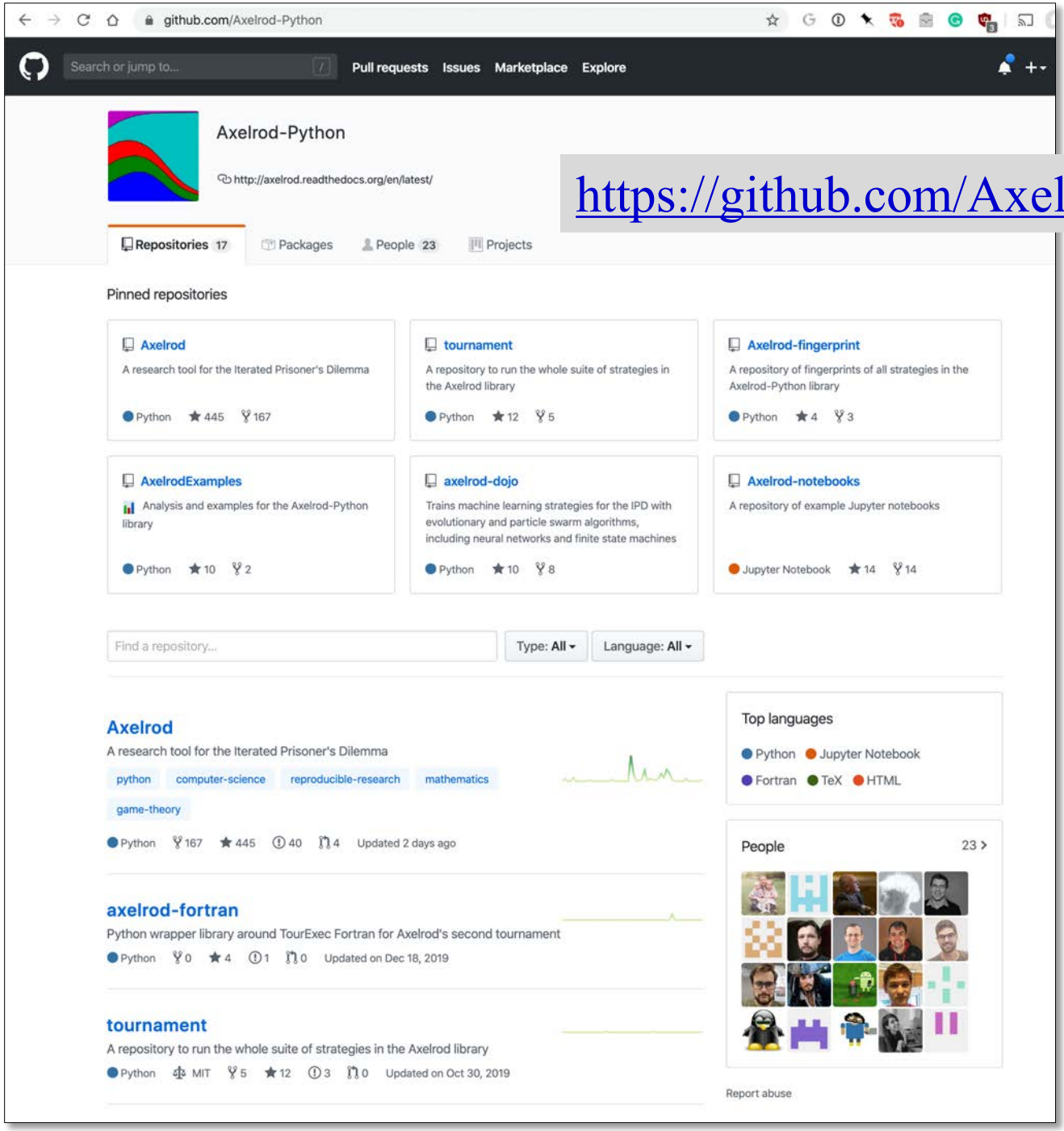
Axelrod

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


github.com/Axelrod-Python
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Axelrod-Python
http://axelrod.readthedocs.org/en/latest/
https://github.com/Axelrod-Python
Repositories 17
Packages
People 23
Projects
Pinned repositories
Axelrod
A research tool for the Iterated Prisoner's Dilemma
Python 445 167
tournament
A repository to run the whole suite of strategies in the Axelrod library
Python 12 5
Axelrod-fingerprint
A repository of fingerprints of all strategies in the Axelrod-Python library
Python 4 3
AxelrodExamples
Analysis and examples for the Axelrod-Python library
Python 10 2
axelrod-dojo
Trains machine learning strategies for the IPD with evolutionary and particle swarm algorithms, including neural networks and finite state machines
Python 10 8
Axelrod-notebooks
A repository of example Jupyter notebooks
Jupyter Notebook 14 14
Find a repository...
Type: All
Language: All
Axelrod
A research tool for the Iterated Prisoner's Dilemma
python
computer-science
reproducible-research
mathematics
game-theory
Python 167 445 40 4 Updated 2 days ago
axelrod-fortran
Python wrapper library around TourExec Fortran for Axelrod's second tournament
Python 0 4 1 0 Updated on Dec 18, 2019
tournament
A repository to run the whole suite of strategies in the Axelrod library
Python MIT 5 12 3 0 Updated on Oct 30, 2019
Top languages
Python
Jupyter Notebook
Fortran
TeX
HTML
People 23 >
Report abuse

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

TitForTat

```
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
```

Remembers only last interaction with a given player

Note use of type hints, added in 3.5

TitFor2Tats

```
class TitFor2Tats(Player):
    """player starts by cooperating and then defects only after 2defects 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
```

Remembers last2 interactions with a given player

Cooperates unless this opponent defected the last two times

Bulley

```
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