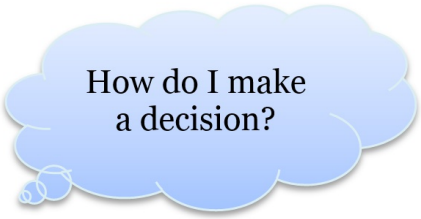
PLTW COMPUTER SCIENCE

Project 1.3.10

Game Theory

Introduction

How do people make decisions? How do rewards, punishments, and ethical and moral beliefs affect people's decisions? How do decisions made in one situation apply to another?.



In this project, you will create a decision-making strategy using *Python*. Your strategy will compete against other teams' strategies in a round-robin tournament of a game called the Iterative Prisoner's Dilemma (IPD). The IPD is a fundamental problem in game theory. Social scientists can use game theory to describe and predict people's behavior. Examples include economic phenomena like stock market fluctuations and political phenomena like revolutions. Nations use game theory to simulate the outcomes of various options when negotiating economic treaties or conducting war.

Materials

- Computer with Enthought Canopy distribution of *Python*® programming language, Internet browser, and GitHub client software
- GitHub individual account and membership in organizational account's team

Procedure

Part I: Describe a decision-making algorithm

1. Form teams of two as directed by your teacher. Meet or greet each other to practice professional skills. Set team norms.
2. Game theorists use algorithms to describe people's decisions. Consider your decision to buy lunch each day. If the cafeteria director decides to raise the lunch price above a certain point, you will opt not to buy school lunch. At what price? Are there other factors? Express your algorithm for the decision to buy school lunch in terms of price and other factors. You could use any language you care to, but do not use the computer. Your expression must be conveyed in writing or through a graphical representation.
3. The Iterative Prisoner's Dilemma is a series of rounds of the Prisoner's Dilemma**.** The Prisoner's Dilemma was one of the early algorithmic problems in game theory, first posed in 1950. In the Prisoner's Dilemma, you and another person have committed a crime together,

and you are caught without evidence. The police question you and the other person separately.

- If both of you collude together and refuse to talk the police, you both will go free.
- If you both confess your crime and betray each other, the liability of the crime will be split between you. You both will receive the standard punishment.
- If you stonewall the police, hoping to collude with the other person but that person betrays you, you will receive an unusually severe punishment while they go free and get to keep the stolen goods.
- If you betray the other person while they attempt to collude with you, you will be set free and get a cash payment

  According to the table below, review your possible results*, o*ne of **R**elease, **T**reat, **S**evere punishment, or **P**unishment:

		Your Action	
		Collude	Betray
Other Player's Action	Collude	**R**eleased: *0 points*	Set free with **T**reat: *+ 100 points*
	Betray	**S**evere punishment: *– 500 points*	**P**unishment: *– 250 points*

What is the maximum number of points possible for you in one round of this game? When does it occur?

4. Face off against another player to play a few rounds of the IPD without a computer. On the count of three (you can chant “IPD” together,) each player reveals their decision to collude or betray using an open hand or a closed fist.

5. In the Iterative Prisoner's Dilemma, many rounds are played, one after another, with the same other player. Betraying in one round will sometimes cause the other player to lose trust and be less likely to collude with you in future rounds. Once players start betraying each other, round after round they end up turning each other in and doing poorly. Considered together with the other player, your outcomes are better if you collude round after round

   The Prisoner's Dilemma lets social scientists study the conditions under which people will either act selfishly or act in the collective best interests of the group. Think of one situation in which a person in your life could act selfishly or in the interest of a common group, and explain how this relates to the Prisoner's Dilemma

6. Our recent capacity to generate, collect, store, and analyze huge amounts of data quickly has caused dramatic changes in all career fields. Simulation was an unimportant tool in social sciences in 1960. Explain why the rise in computational power has changed the career fields in social sciences such that simulation is now a fundamental tool for many social scientists.

Part II. Explore the Simulation

or switch to another branch besides the master. In the GitHub Client window, note
rd “master” with an arrow next to it. That indicates you are working on the master
. The master branch is what we will eventually make our changes to in order to submit
equest to merge our code with the rest of the class. For now, we want a “testingAB”
where “AB” are your initials. This branch will allow us to explore and make changes to
le without affecting the master branch. Select the triangle indicating a dropdown menu
oe “testingAB” in the text box. Select the checkmark to create a new branch. Note that it
ys “testingAB” where before it said “master.” That means you are on the testingxx
, not the master branch.

risonersDilemma.py in the *Python* code editor.

e the code, which imports prisoners’ dilemmaplayer algorithms from other files and
several functions for running a tournament. The code also simulates a tournament
four of the imported strategies called example0 (“E0”), example1 (“E1”), example2
and example3 (“E3”). Data from a round-robin tournament among four players each
one of these strategies is printed on screen and stored in a data file. Scroll up to
ne the three sections of printed output.

```
------------------------------------------------------------------------
on 0 - Line up
------------------------------------------------------------------------
r 0 (P0): E0, Collude
   Always collude.
r 1 (P1): E1, Betray
   Always betray.
r 2 (P2): E2, Alternate
   Collude, then alternate.
r 3 (P3): E3, Collude but retaliate
   Collude first round. Collude, except in a round after getting
   a severe punishment.
------------------------------------------------------------------------
on 1 - Player vs. Player
------------------------------------------------------------------------
column shows pts/round earned against each other player:
        P0     P1     P2     P3
0 :      0    100     50      0
1 :   -500      0   -376   -375
2 :   -250    -75      0   -199
3 :      0    -75   -199      0
  :   -750    -50   -525   -574
```

```
--------------------------------------------------------------------------------
on 2 - Leaderboard
--------------------------------------------------------------------------------
ge points per round:
name (P#):  Score      with strategy name
     (P1):        -13 points with Betray
     (P2):       -132 points with Alternate
     (P3):       -144 points with Collude but retaliate
     (P0):       -188 points with Collude
```

riable `reports` contains these report sections. There is also a Section 3, containing a te report for each team. The following command at the IPython prompt will display the n 3 report for player O.

```
: reports[3][0]
```

IPython prompt, execute the following commands and examine the output.

```
: scores, moves, reports = main_play([example1, example2, team0])

: scores, moves, reports = main_play([example1]*3 + [example2]*2)
```

nction `main_play()` takes a single argument: a list. Explain the purpose of that list, on your examination of the program's output. Note that multiplying a list and an int es a list with repeated elements:

```
ple1]*3 + [example2]*2
valent to
ple1, example1, example1, example2, example2]
```

ne data about past moves. The simulation program represents each decision by one wo characters `'c'` or `'b'` to collude or betray. Before making the decision each round, layer (or algorithm) can consider what has happened in previous rounds. The ms have the previous rounds' information in the form of a string. For example `'ccb'` es that the player colluded in the first two rounds and betrayed in the most recent round. layer can consider two strings: one for their own history and one for their crime 's history. Consider the following two histories:

istory	`'cccccc'`
rime partner's	`'cccccb'`

n tell that five rounds have been played already. What should be each player's score, on their moves?

ur players in the simulation shown above used four different algorithms. The *Python* nplementing the algorithms strategies is shown below. Only the algorithm in Example 3 m considers what has occurred in previous rounds. It uses `my_history` and `_history`, which are strings as described in Step 14.

```
mple 0. Always collude
ove(my_history, their_history, my_score, their_score):
eturn 'c'

mple 1. Always collude
ove(my_history, their_history, my_score, their_score):
eturn 'c'

mple 2. Collude, then alternate
ove(my_history, their_history, my_score, their_score):
f len(my_history)%2 == 0:
   return 'c'
lse:
   return

mple 3. Collude but retaliate
ove(my_history, their_history, my_score, their_score):
f len(my_history)==0: # It's the first round; collude.
   return 'c'
lif my_history[-1]=='c' and their_history[-1]=='b':
   return 'b' # Betray if severely punished last time,
lse:
   return 'c' # otherwise collude
```

ound match between player O and player 1 results as follows:

›le 0:	'cccccccccc'	Score = -5000
›le 1:	'bbbbbbbbbb'	Score = +1000

e the code for these algorithms and record the results you expect from a ten-round of the IPD between example2 and example3, following the previous example matching le0 to example1.

the output. Run a tournament that includes example2 and example3 among the ies. Open the file`tournament.txt`. The simulation program will have stored that file in ne directory as the simulation program itself. The file probably opens by default in the ıtion Notepad, but any text editor will do. At the top of the file is the record for team 1 vs. , showing their final scores per round, the names for their strategies, and a record of ecisions in b and c strings. When the simulation runs, it runs 100 to 200 rounds of the ıa between each pair of strategies.

```
1 vs. team 0
s. -500
tabber vs. loyal
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbb
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccc
```

down until you find the data stored for example2 vs. example3. Is it different than your ion? If so, how

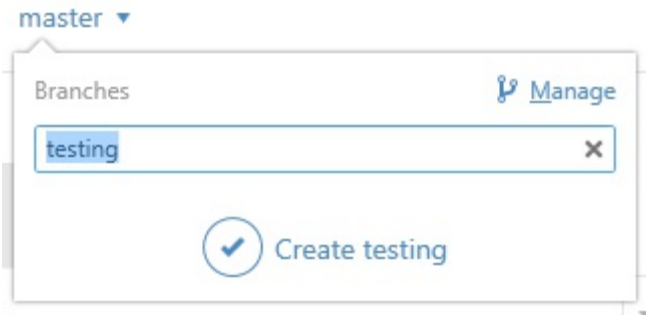
that a strategy's success is dependent upon the strategies it is collaborating with and ting against. Discuss this with one or more other teams in the class and describe why curs.

Part III: Develop Your Own Algorithm

16. You will now develop several algorithms to test against each other. Eventually you will select one algorithm to contribute to the classroom tournament. That one algorithm can integrate all of your initial algorithms or other algorithms. Your final algorithm must call at least one other procedure that you create. For example, your function move could use another function that you define called `probability_that_other player_will_collude`. With your partner, brainstorm algorithms for making the collude-vs.-betray decision. Recall the ground rules for brainstorming:

    - Quantity over quality
    - No evaluation of ideas, but piling on is okay
    - Record ideas with tag lines of one or a few words

17. Review your team norms for the driver and navigator roles in pair programming.
18. Prepare to develop. Pick three of your ideas to implement in the *Python* code in the simulation program. Each idea will be implemented in a different teamXX file. Describe your ideas in human language and pseudocode.



19. Implement your ideas with code. Use test-driven development as follows
    -

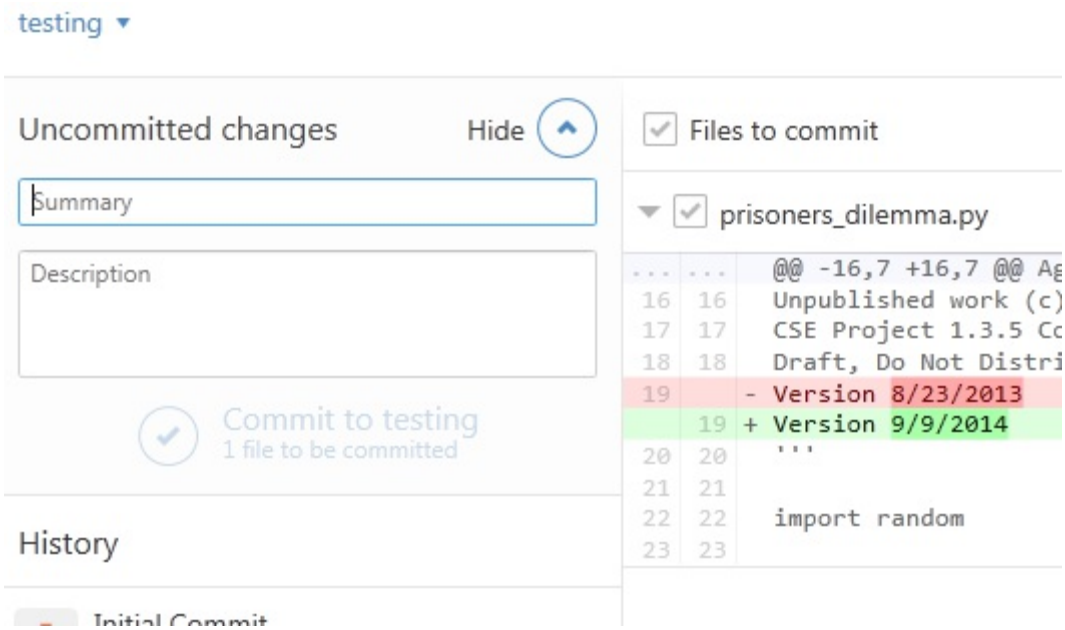
Create a test. In Canopy, open the team file assigned to you by your teacher. The `teamNN.py` file is in the folder for your local repository. Think about how you could develop your algorithm in stages. What is the first stage? In the code at the end of the teamXX file, record whether your code at the end of that first stage should return ‘b’ or ‘c’ for one particular set of the four arguments. For example, the following test checks whether `move()` returns ‘b’ on the first round.

```
# Test 1
if test_move(my_history='',
             their_history='',
             my_score=0,
             their_score=0,
             result='b'):
    print 'Test passed'
```

- Pair programming, create code that will pass this test. Execute and revise your code until it passes your test(s).

20.

Commit the changes. You saved your changes on your computer when you executed the file. However, you did not commit those changes to the cloned repo on your computer. In the GitHub Client, select the Changes tab. You should see uncommitted changes to the repository. Select one of the files that has been changed and you should see lines you deleted (red) or added (green).



- Although your changes are saved on your local computer, they are not stored permanently by git. To create a permanent checkpoint of this change, you need to make a commit. Enter a summary (e.g., “Make first move”) and description (e.g., “Always collude on first move. Also created a test to check this.”) and select **Commit to testingAB**.
- You now have a record of that change in the testing branch. However, the master branch is still unchanged. Click on the arrow next to testing and switch back to the master branch. You will see that there are no changes to that branch. When you switch to the master branch, Canopy may warn you that the file has changed on disk; the change you made has disappeared! Git actually changed the file on the computer to reflect what is current in the master branch. The change still exists – try changing back to the testing branch. You can see how you can make changes to code without affecting the master version of the code and easily switch between the two.
- Go back to and stay on the testing branch. Repeat the process as you develop within the testing branch
    - Plan a small task.
    - Create a test that will check that your code correctly accomplishes the task you plan.
    - Creating code to pass the test.
    - Commit the successful code to the testing branch.

21. **Create a program.** Implement three strategies of your design, one strategy in each of three files. Run tournaments in which you observe how your strategies do in competition and

collaboration with other possible strategies. Record the results of your simulations. Complete the following for each idea:

- Write pseudocode and strategize how you will implement the algorithm in *Python*.
- Implement your algorithm in Python in one of the team files. Include `team_name`, `strategy_name`, and `strategy_description`.
- Examine the `tournament.txt` file or the Section 4 reports to see how your algorithm performed against each other strategy.
- Record notes about your thinking as you develop your algorithms further. In your notes, include the performance of your algorithm against other strategies:

They Scored \ You Scored	Your algorithm #1
Always collude	
Always betray	
Use their previous move	

22. Decide on one algorithms to contribute to the whole-class tournament. Explain your reasoning for choosing that algorithm.

Part IV. Contribute Your Strategy to the Class Collaboration

23. Highlight the code for the strategy that you are going to use in the tournament and copy it to the clipboard (Edit → Copy or Ctrl+C).
24. Switch to the “master” branch of your fork of the IPD Repository:
    - In the GitHub for Windows Client, Click on the arrow next to “testing” and choose “master.”

- Canopy will automatically update the code file you have open, showing the original version of the code.
- Paste the copied code into the file designated for your team. Be careful not to make any changes in any of the other files.

25.

You should have your IPD strategy in the Python file within the master branch. Once you save the Python file you will need to make a commit in the GitHub for Windows Client. Sync your repository so the remote and client are current.

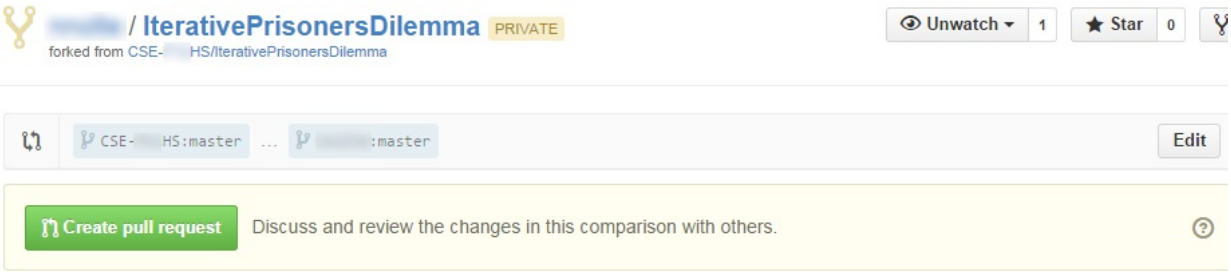
26.

Open the repository on GitHub by either right-clicking on it in the list or directing a Web browser to github.com.

27. Create a **Pull Request** so your changes can be merged into the original code.

- Click on the green symbol as shown to enter the compare and pull request screen:

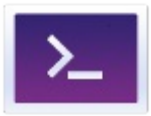
- The top line shows what two things you are comparing. In this case it should show the master branch of the original repository in your school's organization account (the base fork) and your master branch in your forked copy of the repository (the head fork). If it does not, click on the “Edit” button to change it.



- Click on “Create pull request” and add in a title a comment (much like a commit message). In the pull request type the names of your groupmembers.Finish by once again clicking “Create pull request.”

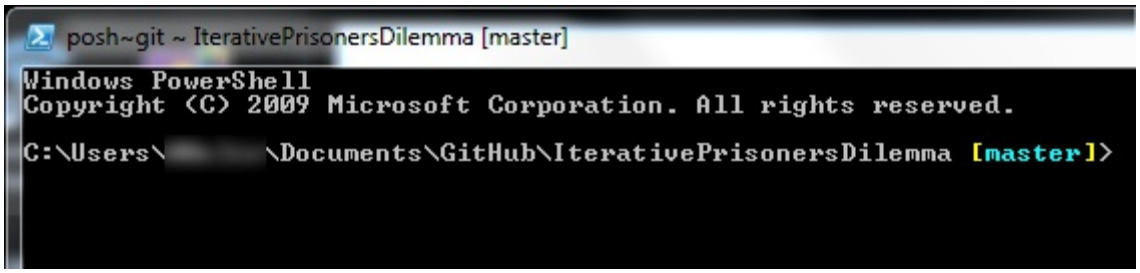
28. Your teacher will now get a message that someone has code they would like to add into the original repository. If everything was done correctly, your class's code will be easily merged into the original class repo's master branch. The current commit in that master branch will have the original code modified with the code of every other team. If two different developers both make changes on the same line of code, merging the commits requires work in the git shell, work which you could learn to do in a later course.

    Was the teacher able to easily merge your code with the code from others merged before you?

29. 

    Before you can run the tournament you need to get all of the changes made to the class' repo by other teams. To do this, you will need to resync the original fork you made. You will need to use the git shell.

    - Open the git shell either by opening the program indicated by the icon or by right-clicking on the repository in the GitHub for Windows Client and choosing “Open in git shell.” A PowerShell window should open, and the path should be the location of your local repository followed by [master].



    - Run the command:

      ```
      git fetch upstream
      ```

This gets the code from the original master branch, which should now have every team's code incorporated. Next, run the command:

```
git merge upstream/master
```

This command updates your fork to be the same as the original repository.

Part V. Run the Iterative Prisoner's Dilemma Tournament

30. Once everyone's code has been merged with the original repo and you have re-synced your fork, you are ready to run the tournament. Make sure you are in the “master” branch of your fork so you have code from all teams. Execute the function and examine the `tournament.txt` file. Who won? How did your strategy do overall?

31. Against which strategies did your strategy perform well? Which strategies defeated you? Explain why the winning strategy or strategies performed well.

32. Write a strategy in pseudocode that would do better when paired with a team that you fared poorly against in the competition. Explain why you think this new strategy would or would not do well overall.

Conclusion

1. Game theory describes any situation in which two or more people have to make a decision without knowing the other person's decision. Usually, the consequences of our own decision depend on other people's decision. So we have to make our decision based on guesses about what the other person will do.

   Describe some situations in which you make a decision based on guesses about other people's decisions.

2. Use the Web to find about how computer simulation has changed the social sciences, including applications in diplomacy, warfare, and economics. Briefly describe your findings.

3. Practice Opportunity for the Create Performance Task.

   Each function you create is an abstraction. “Explain how an abstraction you created helped manage the complexity of your program.” (adapted from College Board Create Performance Task Part 2d.)

   Note: This task does not duplicate the content of the College Board task or rubric. The task provided here contains elements that are different than the College Board Performance Tasks and Rubrics. Please reference official College Board materials.