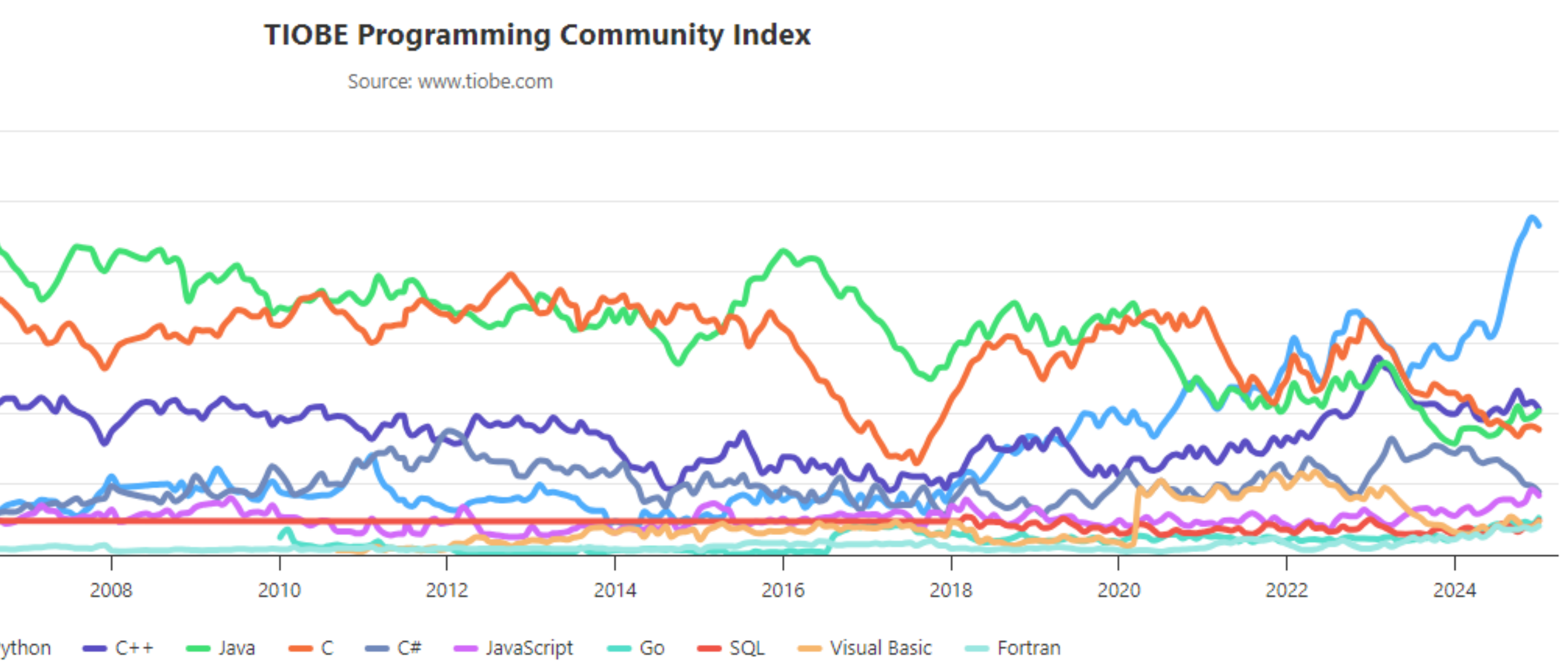
Programming for Social Scientists

Introduction

Johan A. Dornschneider-Elkink


from watson.framework import events
from watson.http.messages import Response, Request
from watson.common.imports import get_qualified_name
from watson.common.contextmanagers import suppress
ACCEPTABLE_RETURN_TYPES = (str, int, float, bool)
Attributes:
__action__ (string): The last action that was called
"""
def execute(self, **kwargs):
method = self.get_execute_method(**kwargs)
self.__action__ = method
return method(**kwargs) or {}
@abc.abstractmethod
python



The TIOBE Programming Community index is an indicator of the **popularity** of programming languages. The index is updated once a month. The ratings are based on the **number of skilled engineers** world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is **not about the *best* programming language** or the language in which *most lines of code* have been written.

Python

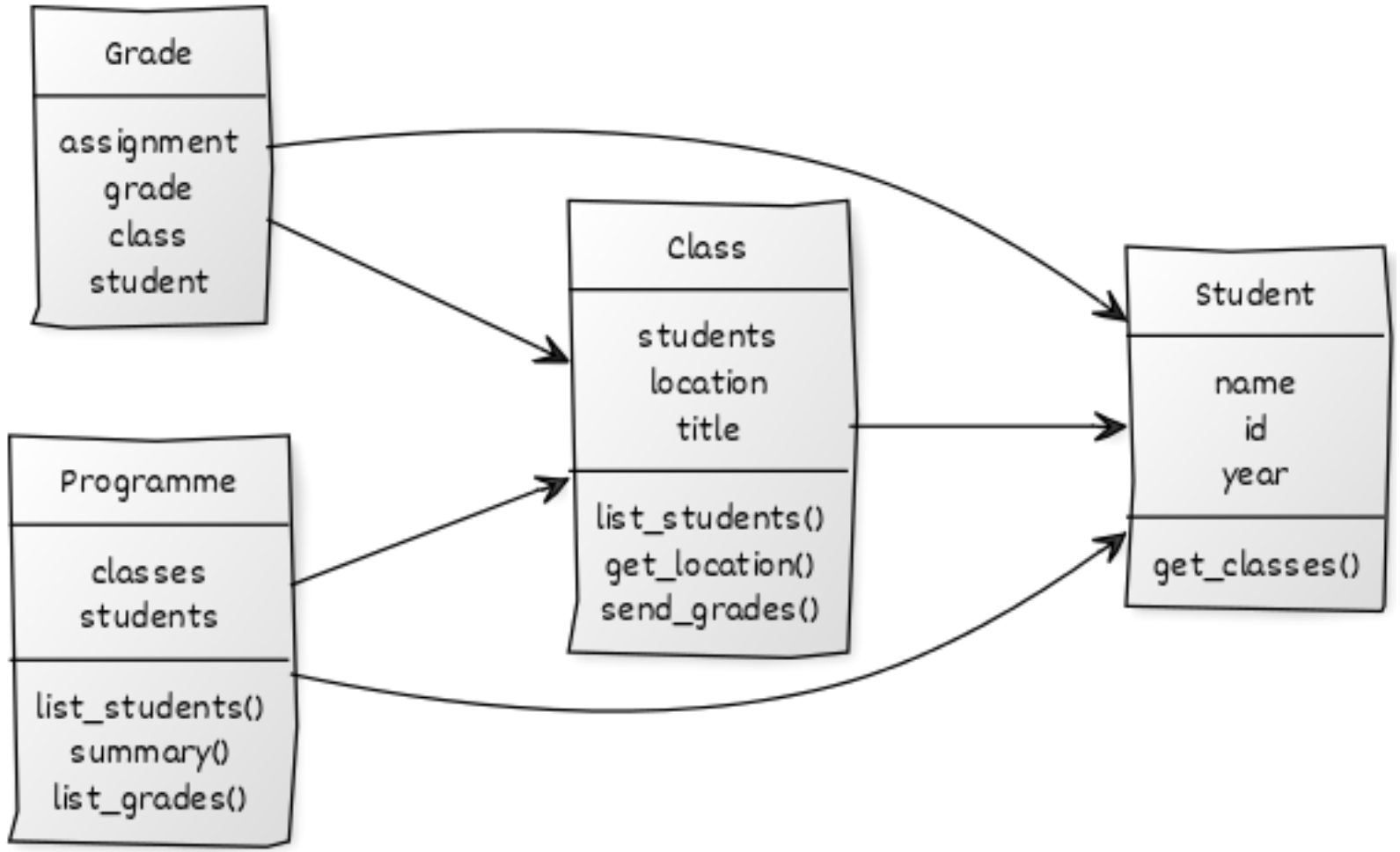
Scripting vs. programming

R vs. python

Applications

Object-oriented design

Object-oriented design



Tools

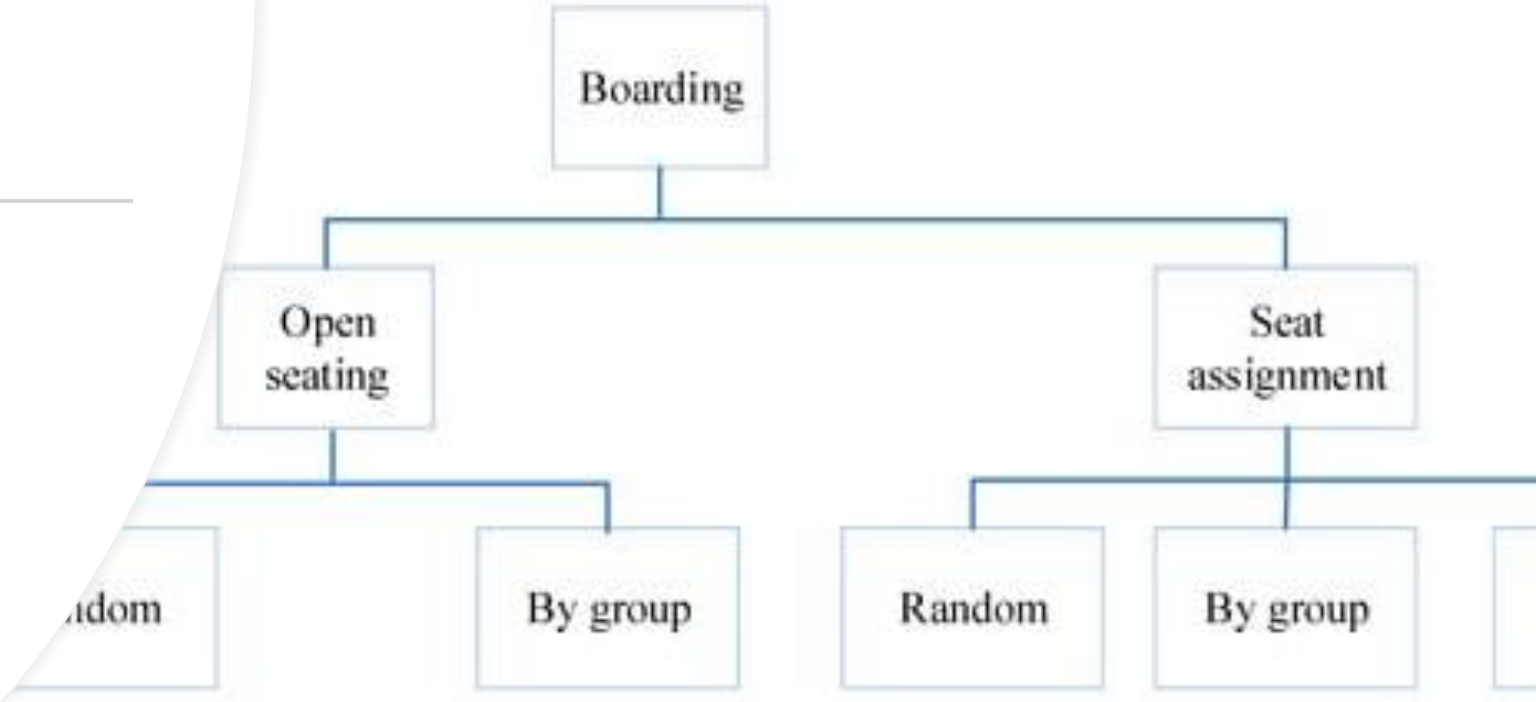
Matrix / Element

Github / git

Repl.it / Visual Studio Code

Local installation ?

Agent-based simulation



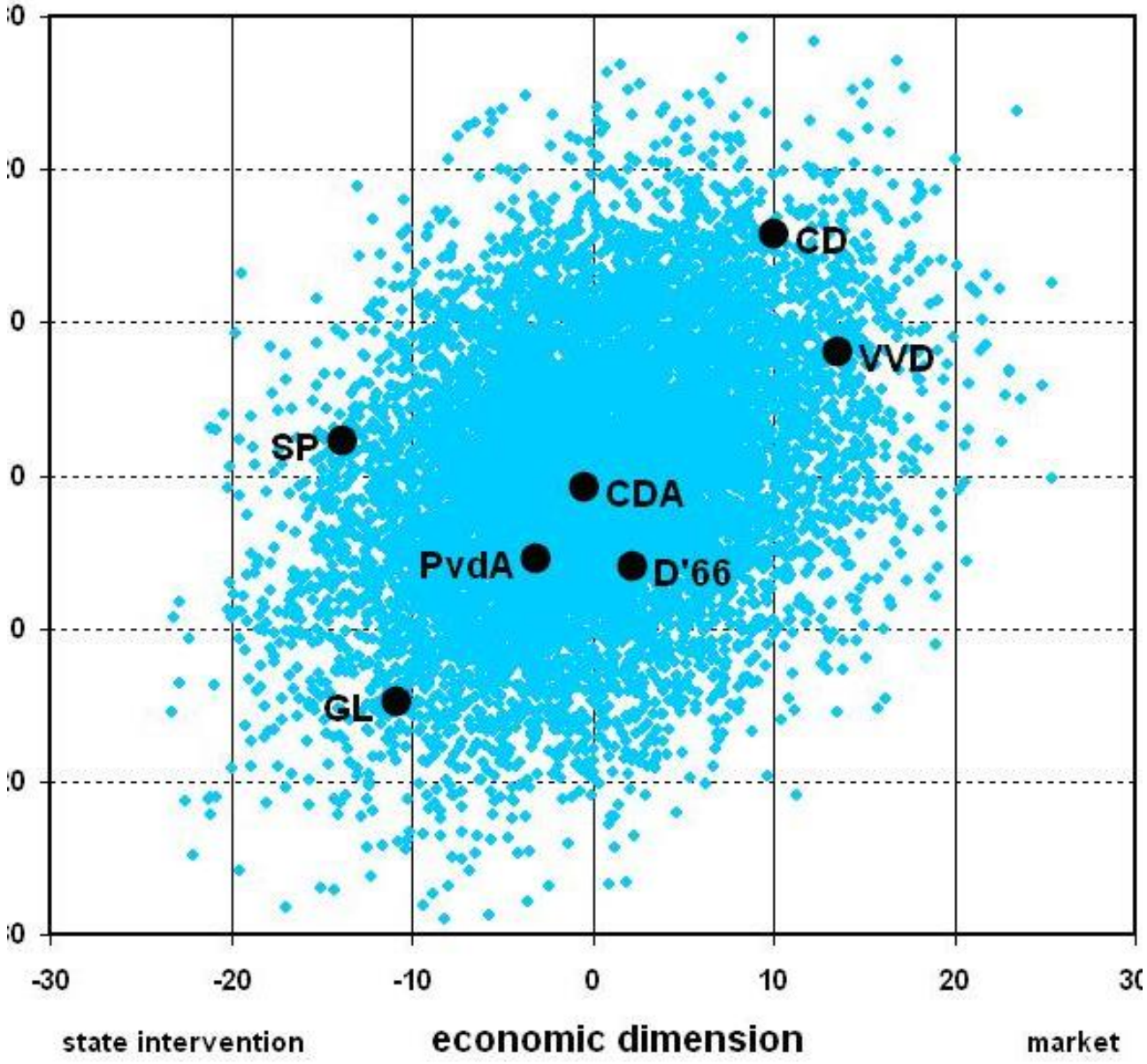
American Political Science Review Vol. 99, No. 2 May 2005

Policy and the Dynamics of Political Competition

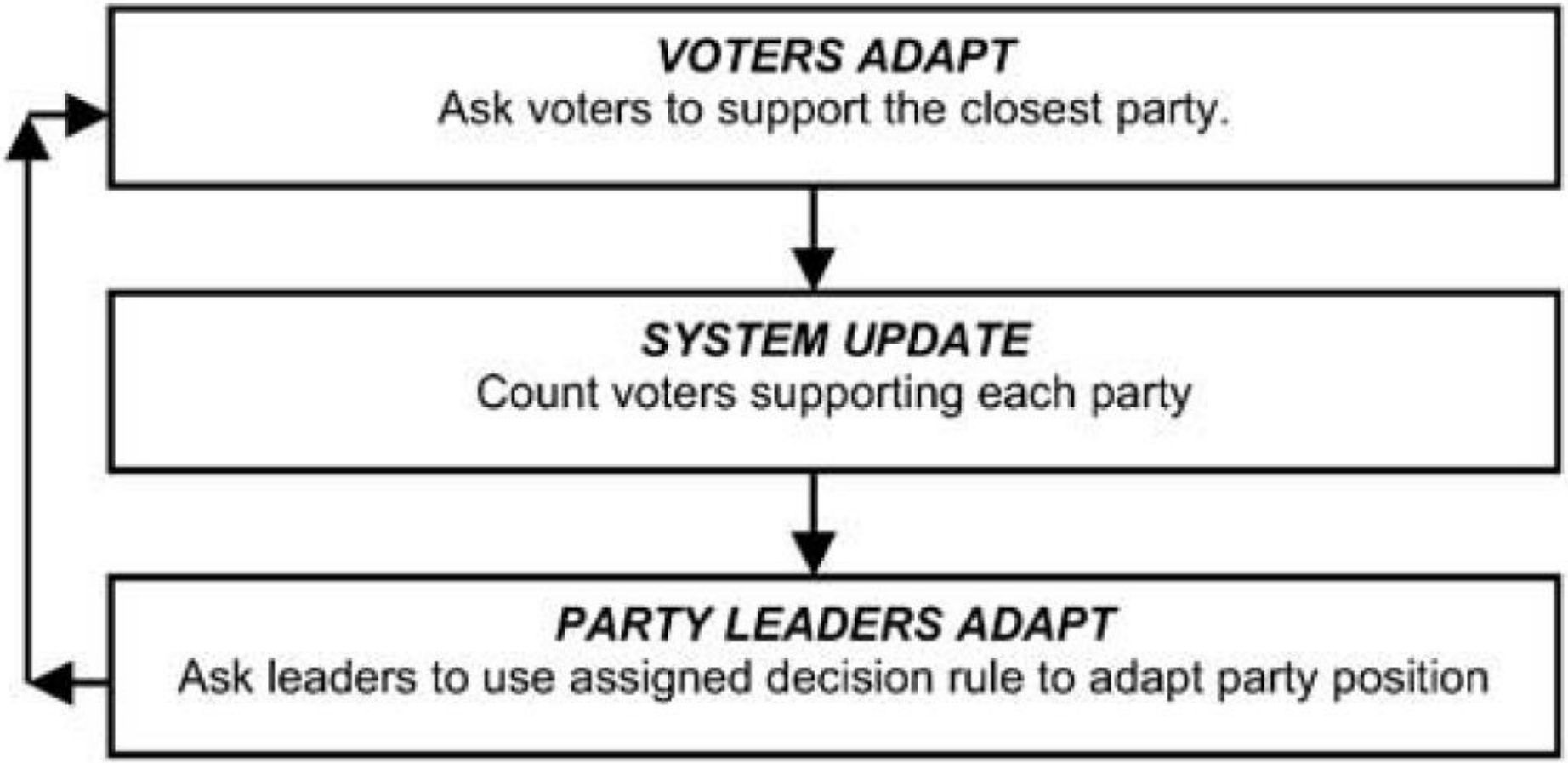
MICHAEL LAVER *New York University*

This paper proposes a model that takes the dynamic agent-based analysis of policy-driven party competition into a multiparty environment. In this, voters continually review party support and switch parties to increase their expectations; parties continually readapt policy positions to the shifting affiliations of voters. Different algorithms for party adaptation are explored, including "Aggregator" (adapt party policy to the ideal policy positions of party supporters), Hunter (repeat policy moves that were rewarded; otherwise make random moves), Predator (move party policy toward the policy position of the largest party), and "Sticker" (never change party policy). Strong trends in the behavior of parties using different methods of adaptation are explored. The model is then applied in a series of experiments to the dynamics of a real party system, described in a published opinion poll time series. This paper reports first steps toward endogenizing key features of the process, including the birth and death of parties, internal party decision rules, and voter ideal points.

Spatial model of voting



Sequence


VOTERS ADAPT
Ask voters to support the closest party.
SYSTEM UPDATE
Count voters supporting each party
PARTY LEADERS ADAPT
Ask leaders to use assigned decision rule to adapt party position

Party strategies

ADAPTIVE DECISION RULES

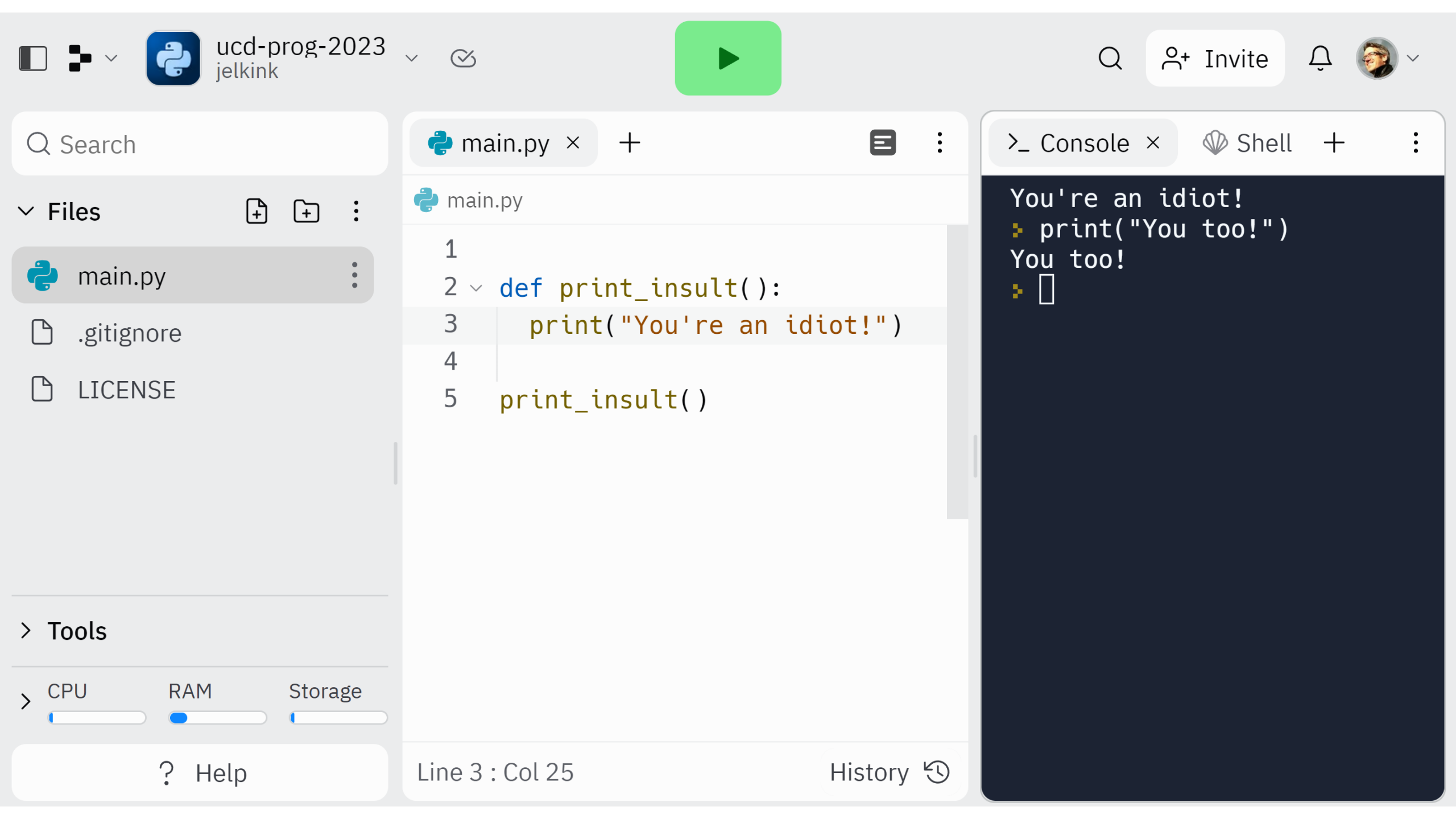
AGGREGATOR
Go to mean position of current party supporters on each dimension.

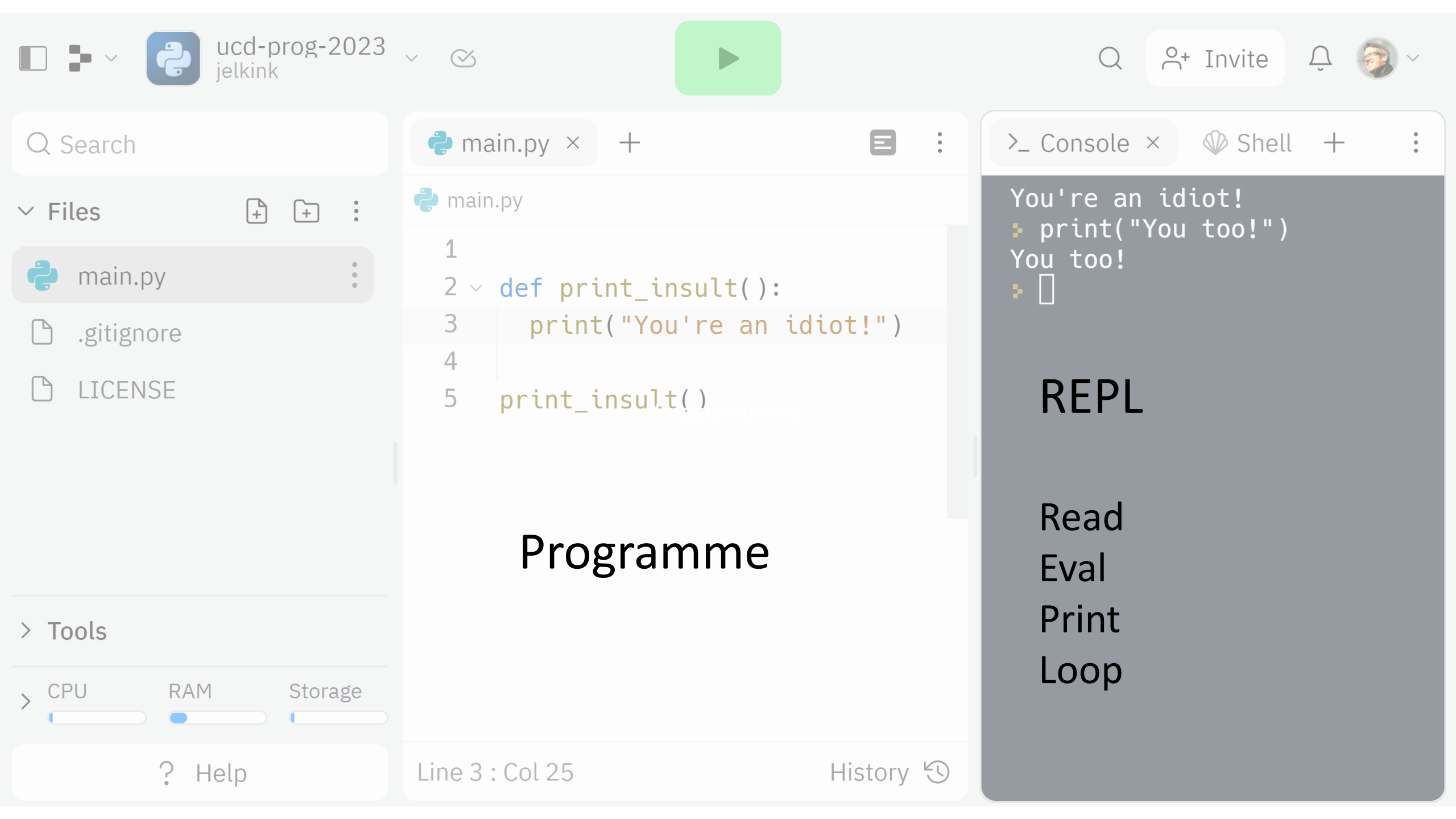
HUNTER
Was previous move followed by increased party support? If yes, repeat move.
If no, turn 180° from direction of last move, make unit move in direction randomly selected from arc 90° either side of direction now faced.

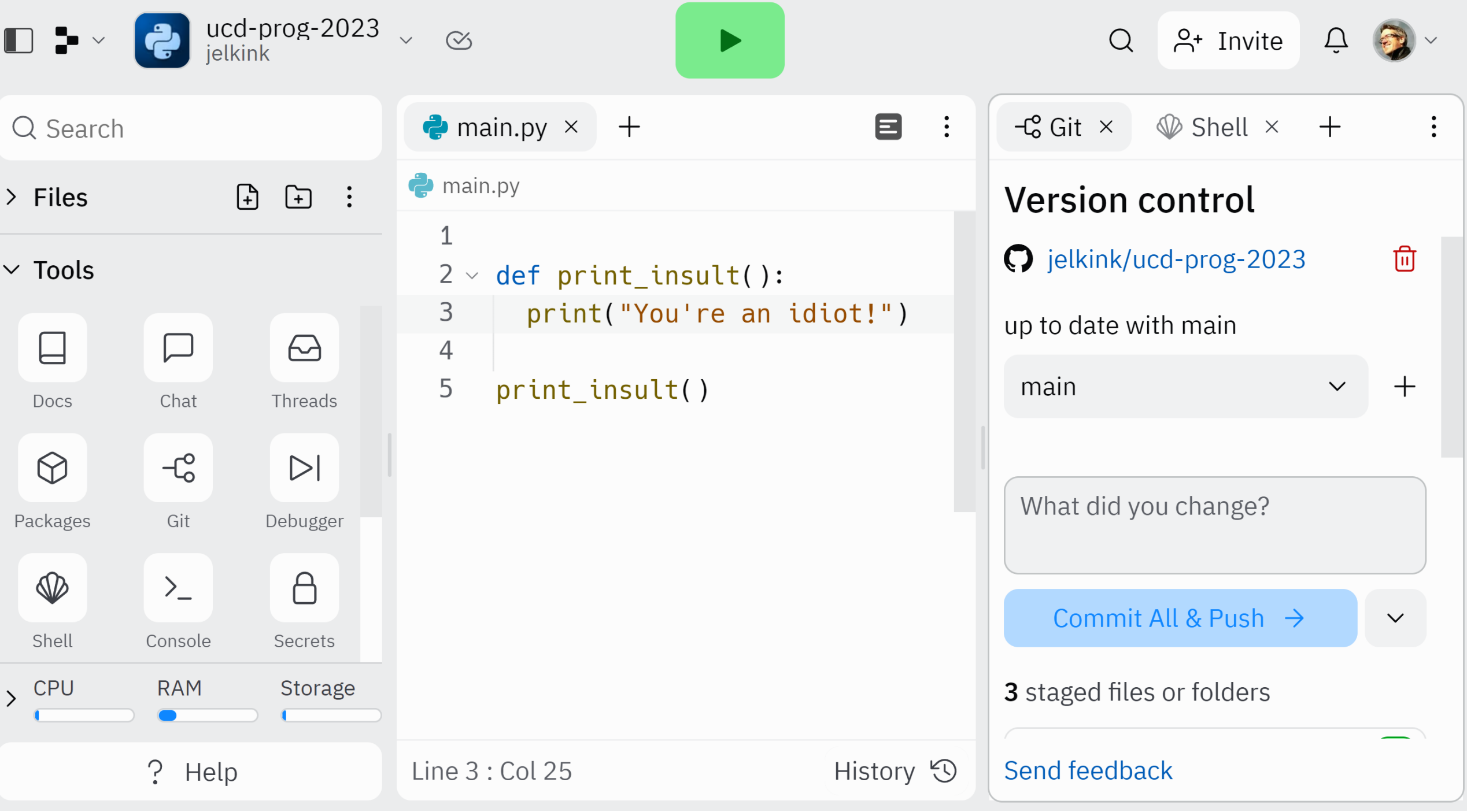
PREDATOR
Observe party sizes. If you are the largest party, stand still.
If not the not largest party, set heading towards largest party, make unit move.

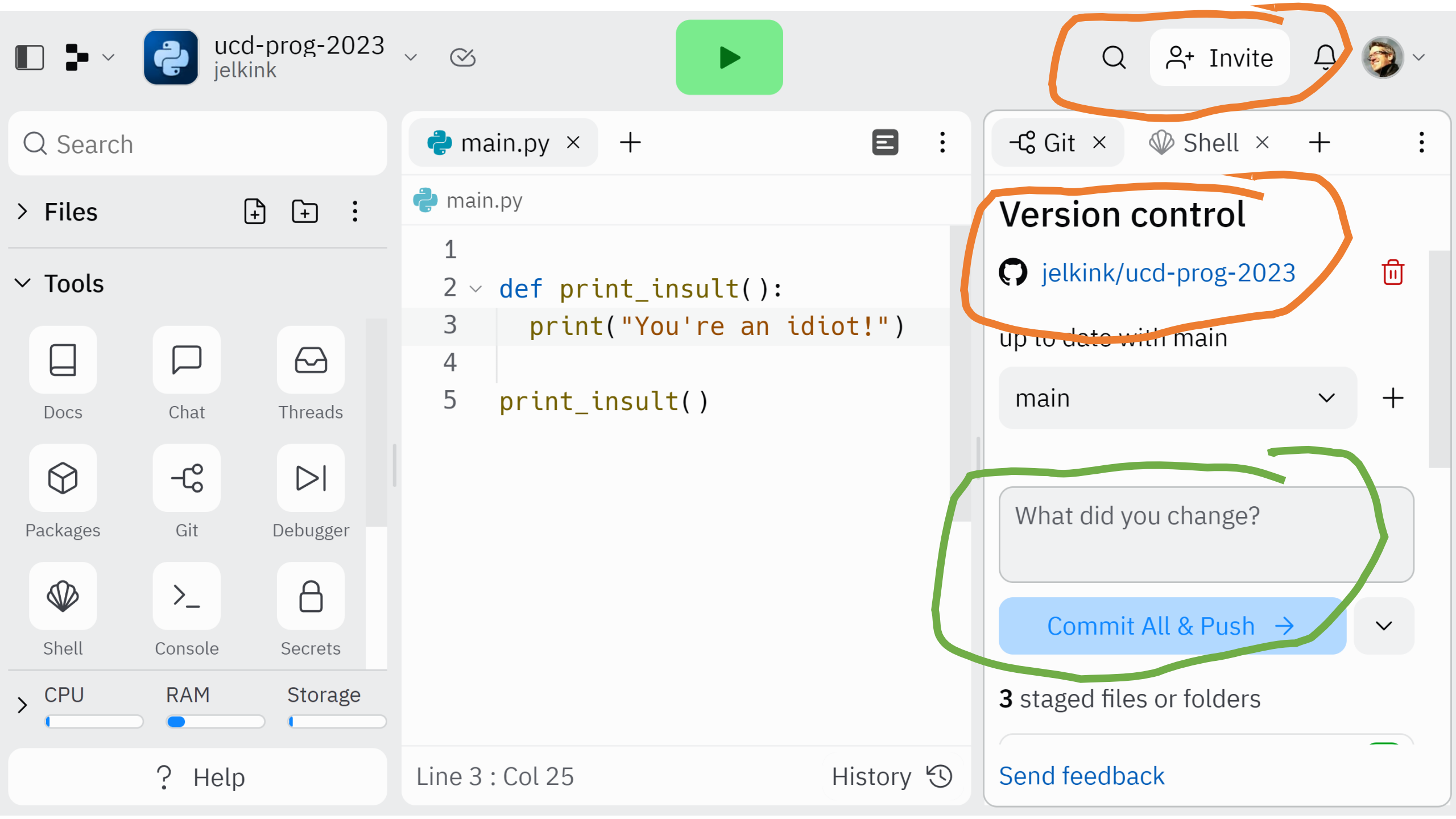
STICKER
Never change policy position.

MCQ test 1	21 Feb	20%
MCQ test 2	28 Mar	20%
MCQ test 3	25 Apr	20%
Class diagram	28 Mar, 1 pm	10%
Lab report	8 May, 5 pm	30%


ucd-prog-2023
jelkink
Invite
Search
Files
main.py
.gitignore
LICENSE
Tools
CPU
RAM
Storage
Help
main.py
main.py
1
2 def print_insult():
3 print("You're an idiot!")
4
5 print_insult()
Line 3 : Col 25
History
Console
Shell
You're an idiot!
print("You too!")
You too!


ucd-prog-2023
jelkink
Invite
Search
Files
main.py
.gitignore
LICENSE
Tools
CPU
RAM
Storage
Help
main.py
main.py
1
2 def print_insult():
3 print("You're an idiot!")
4
5 print_insult()
Programming
Programme
Line 3 : Col 25
History
Console
Shell
You're an idiot!
print("You too!")
You too!
REPL
Read
Eval
Print
Loop


ucd-prog-2023
jelkink
Invite
Search
Files
Tools
Docs
Chat
Threads
Packages
Git
Debugger
Shell
Console
Secrets
CPU
RAM
Storage
Help
main.py
main.py
1
2 def print_insult():
3 print("You're an idiot!")
4
5 print_insult()
Line 3 : Col 25
History
Git
Shell
Version control
jelkink/ucd-prog-2023
up to date with main
main
What did you change?
Commit All & Push
3 staged files or folders
Send feedback


ucd-prog-2023
jelkink
Invite
Search
Files
Tools
Docs
Chat
Threads
Packages
Git
Debugger
Shell
Console
Secrets
CPU
RAM
Storage
Help
main.py
main.py
1
2 def print_insult():
3 print("You're an idiot!")
4
5 print_insult()
Line 3 : Col 25
History
Git
Shell
Version control
jelkink/ucd-prog-2023
up to date with main
main
What did you change?
Commit All & Push
3 staged files or folders
Send feedback

Getting used to Python

```
print("Hello, World!")
```

```
Hello, World!
```

Using Python as calculator

```
5 + 30 * 3
```

```
95
```

```
2 ** 3
```

```
8
```

```
5 / 2
```

```
2.5
```

```
5.0 / 2
```

```
2.5
```

```
5 // 2
```

```
2
```

```
5.0 // 2
```

```
2.0
```

Work on Lab 1

Working with your neighbours is a good idea!