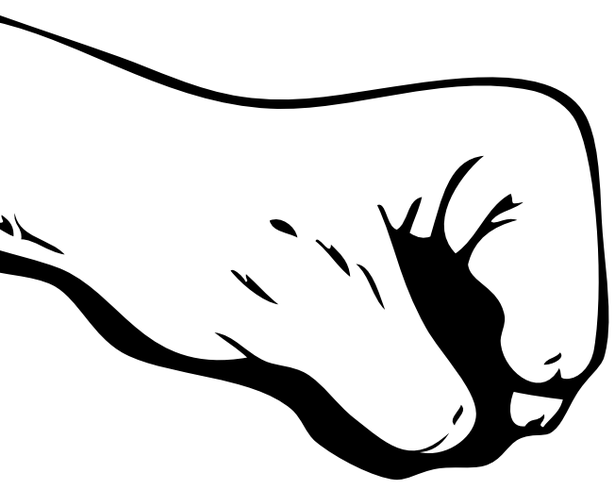
Cheating at



ROCK PAPER SCISSORS

Meta-programming in Python

Matt J Williams
Cardiff University

Django Weekend
7th - 9th February 2014

School of Computer Science & Informatics

Ysgol Cyfrifiadureg a Gwybodeg

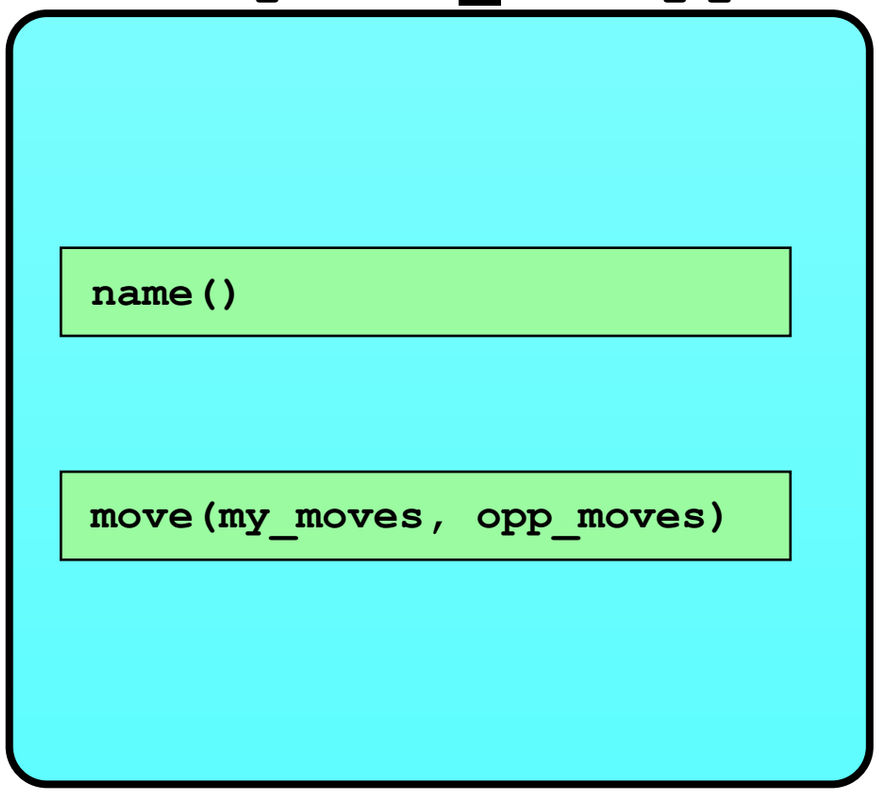
Reception & School Office
Derbynfa a Swyddfa'r Ysgol

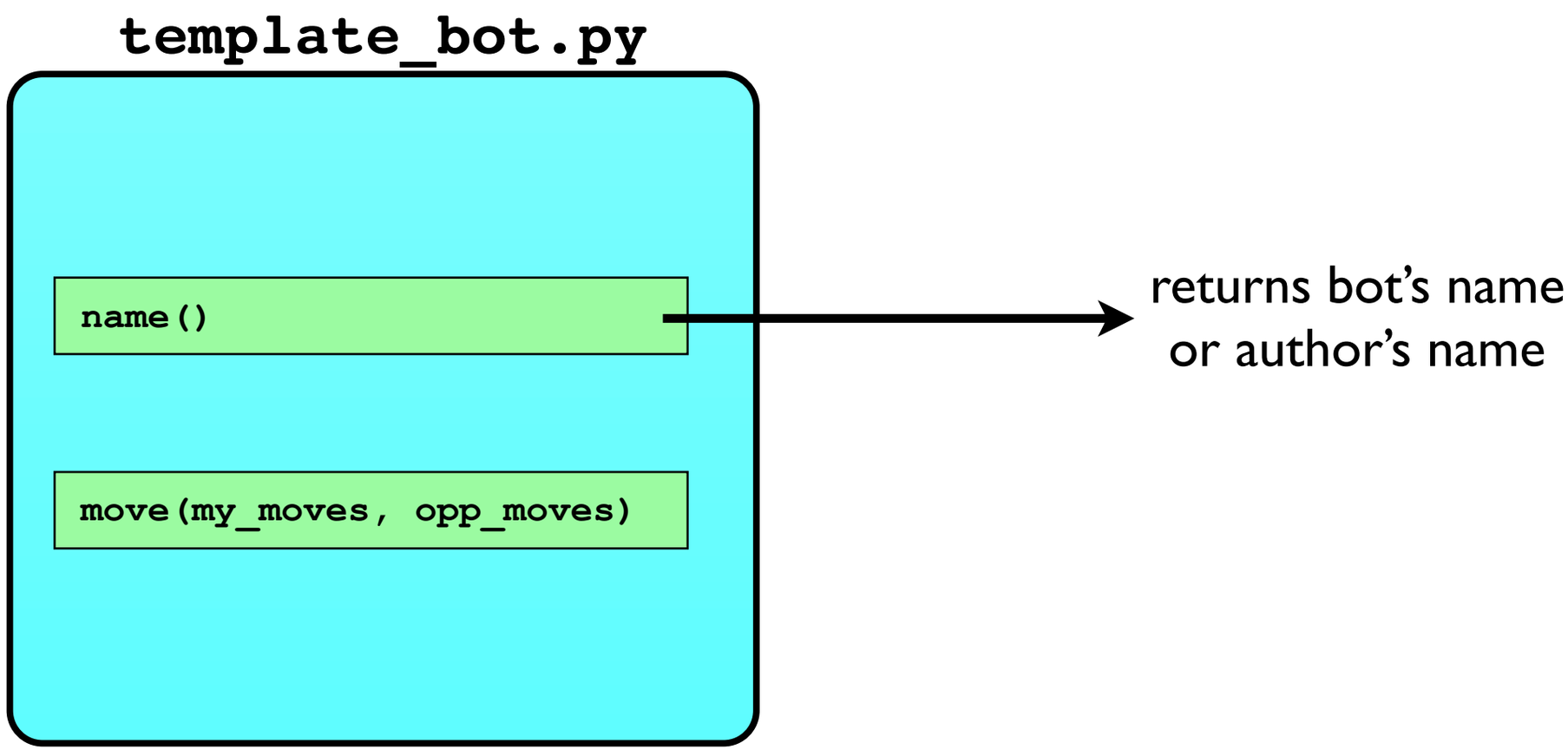
Iterated Rock-Paper-Scissors

- 1,000 rounds of rock-paper-scissors
- Players have a history of their own moves and the opponent's moves

Outcome	Points
Win	3
Draw	0
Lose	1

template_bot.py




template_bot.py
name()
move(my_moves, opp_moves)
returns bot's name
or author's name

template_bot.py

```
name()

move(my_moves, opp_moves)
```

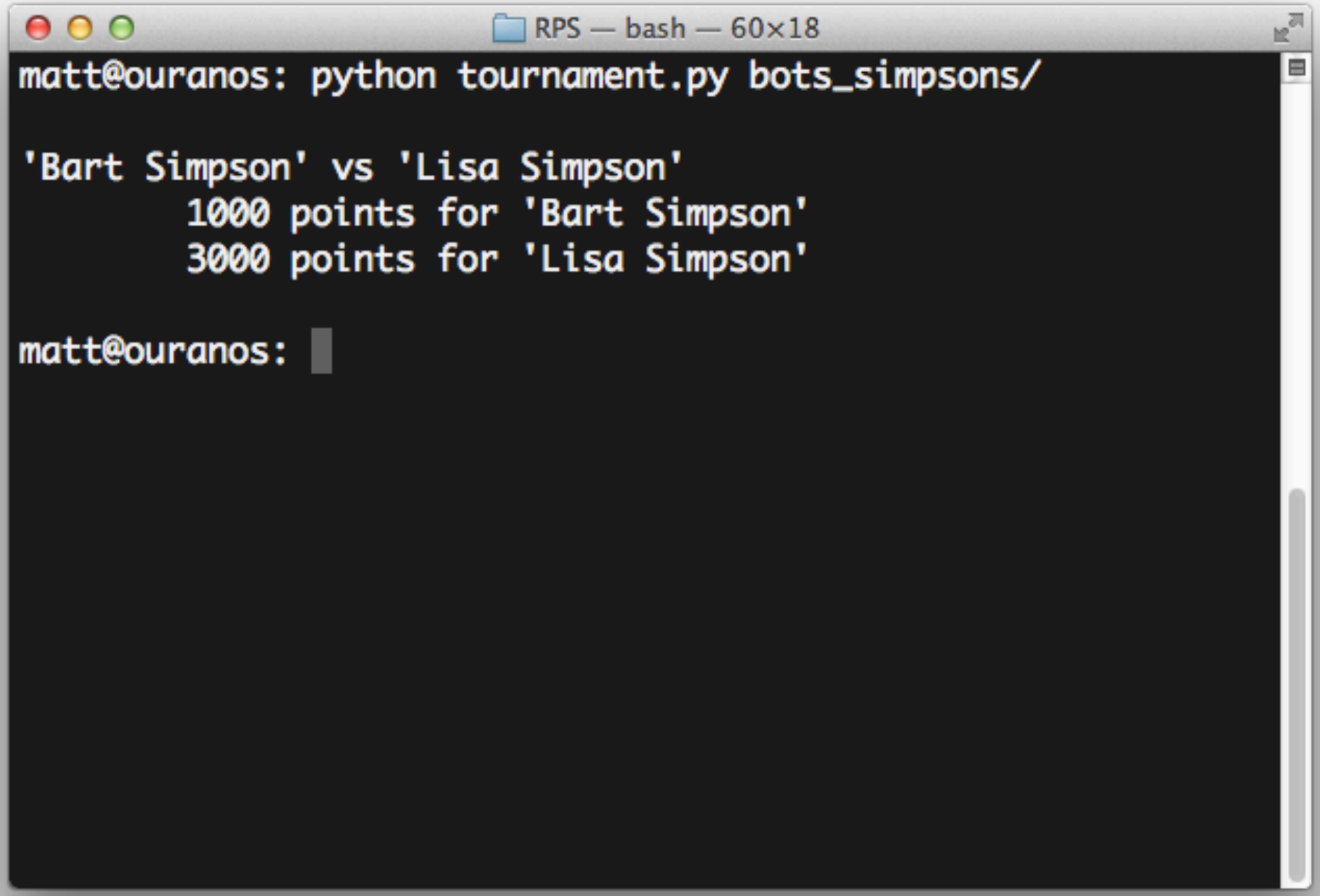
parameters – lists of previous moves
my_moves, **opp_moves**

returns an "**R**", "**P**", or "**S**"

“Good ol’ rock. Rock never fails.”

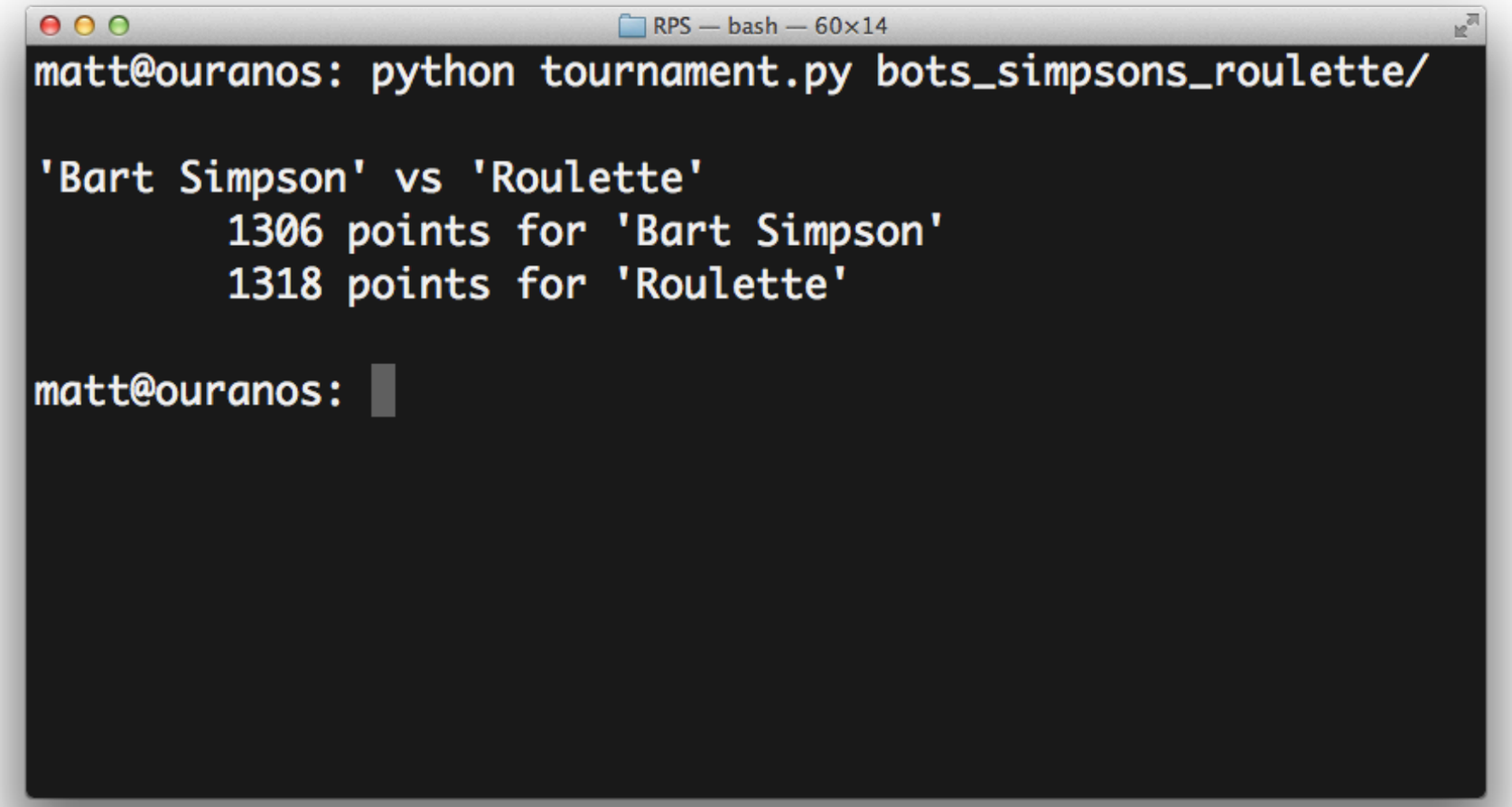
bart.py

```
def name():
    return "Bart Simpson"

def move(my_moves, opp_moves):
    return "R"
```

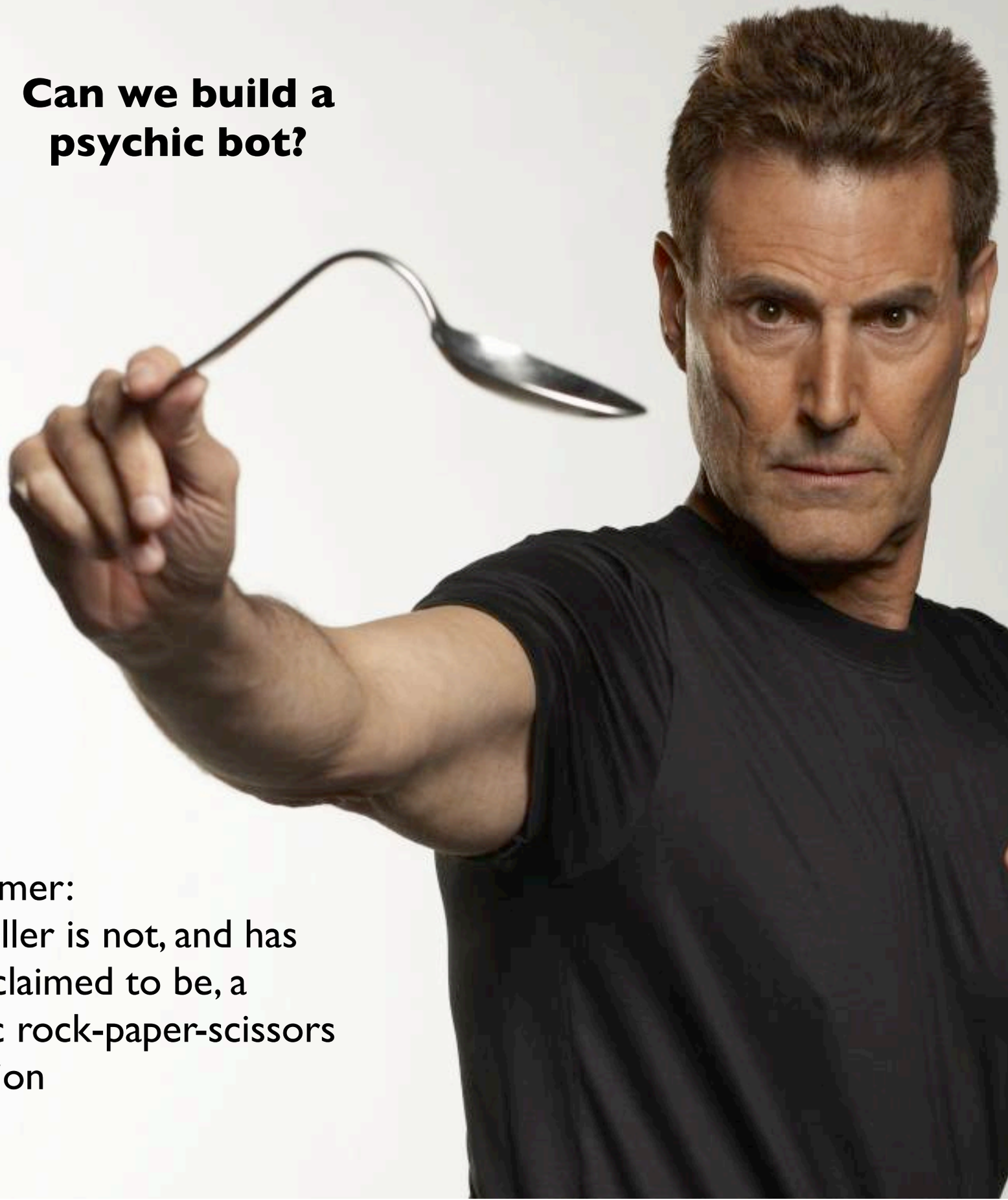

RPS — bash — 60×18
matt@ouranos: python tournament.py bots_simpsons/
'Bart Simpson' vs 'Lisa Simpson'
1000 points for 'Bart Simpson'
3000 points for 'Lisa Simpson'
matt@ouranos:

roulette.py

```
def name():
    return "Roulette"

def move(my_moves, opp_moves):
    return random.choice("RPS”)
```


RPS — bash — 60×14
matt@ouranos: python tournament.py bots_simpsons_roulette/
'Bart Simpson' vs 'Roulette'
1306 points for 'Bart Simpson'
1318 points for 'Roulette'
matt@ouranos:

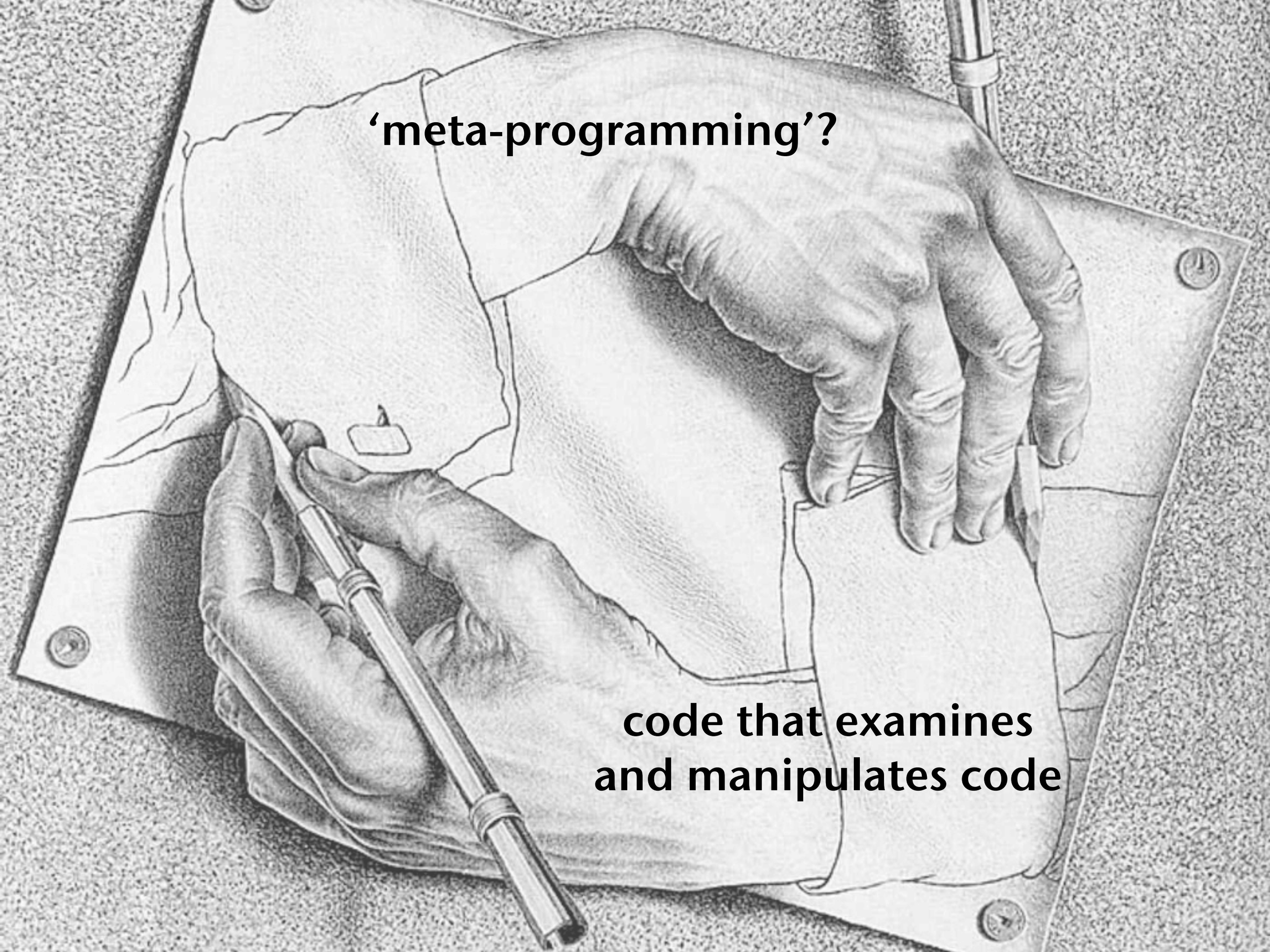
Can we build a psychic bot?

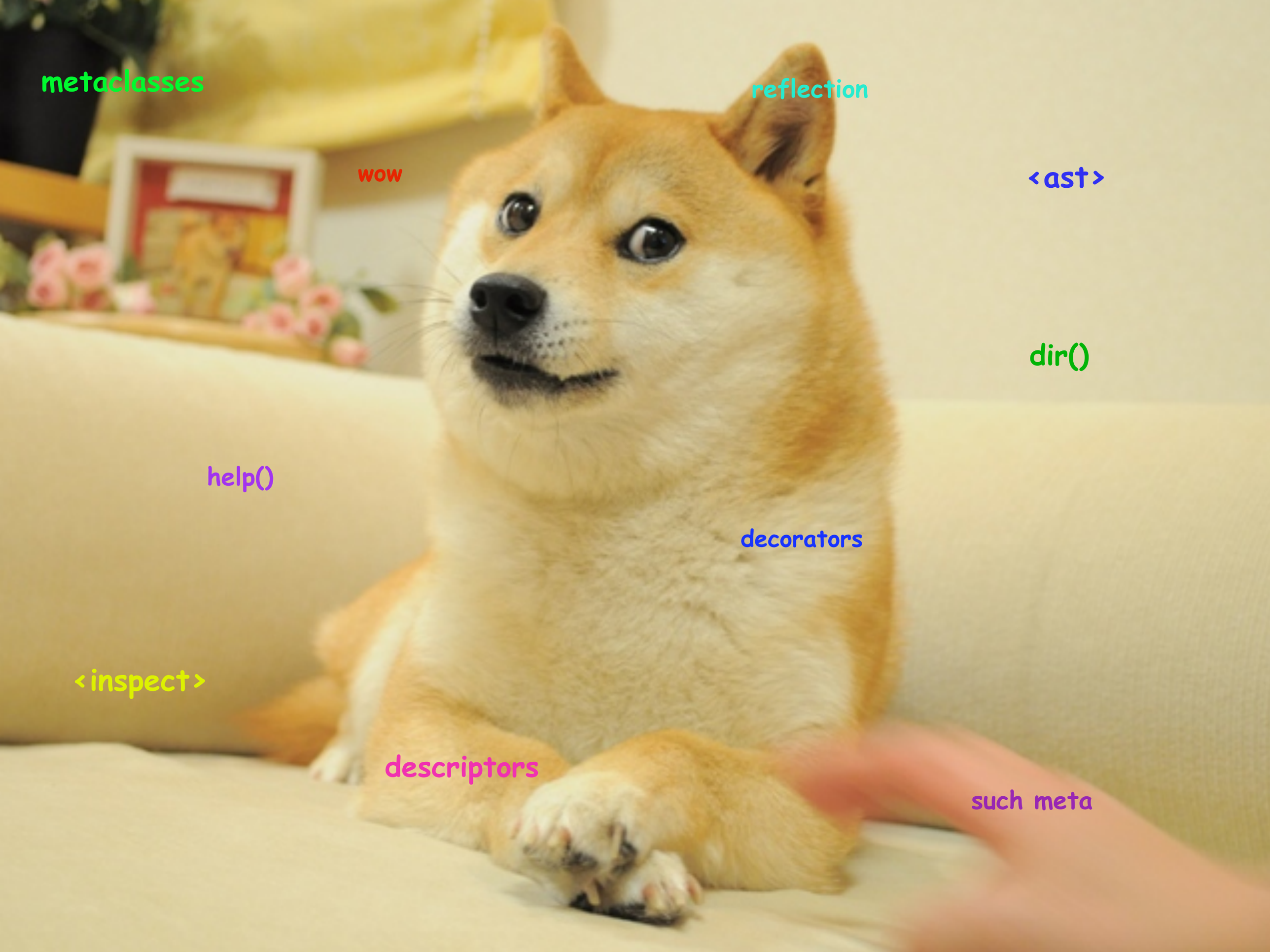


*disclaimer:
Uri Geller is not, and has never claimed to be, a psychic rock-paper-scissors champion

Can our bot...

- **Predict what our opponent will do?**
- **Maybe even *influence* what they'll do?**


‘meta-programming’?
code that examines
and manipulates code


metaclasses
reflection
wow
<ast>
dir()
help()
decorators
<inspect>
descriptors
such meta

27.13. `inspect` — Inspect live objects

The `inspect` module provides several useful functions to help get information about live objects such as modules, classes, methods, functions, tracebacks, frame objects, and code objects. For example, it can help you examine the contents of a class, retrieve the source code of a method, extract and format the argument list for a function, or get all the information you need to display a detailed traceback.

GellerBot Level 1

- Find opponent's *move()* function
- Make them play a test move
- Select the winning move

Locating the opponent's code

tournament.py

```
...

if __name__ == "__main__":
    assert len(sys.argv) == 2
    bots_dir = sys.argv[1]
    assert os.path.isdir(bots_dir)

    #
    # Load modules contained in bots directory
    bots = load_bots(bots_dir)
    num_bots = len(bots)

    #
    # Bots battle each other
    print
    for i1 in xrange(num_bots):
        bot1 = bots[i1]
        for i2 in xrange(i1+1, num_bots):
            bot2 = bots[i2]

            bot1_name = bot1.name()
            bot2_name = bot2.name()
            print "'%s' vs '%s'" % (bot1_name, bot2_name)
            (bot1_points, bot2_points) = battle(bot1, bot2)
            print "\t%d points for '%s'" % (bot1_points, bot1_name)
            print "\t%d points for '%s'" % (bot2_points, bot2_name)
            print

...
```

tournament.py

```
...
if __name__ == "__main__":
    assert len(sys.argv) == 2
    bots_dir = sys.argv[1]
    assert os.path.isdir(bots_dir)

    #
    # Load modules contained in bots directory
    bots = load_bots(bots_dir)
    num_bots = len(bots)

    #
    # Bots battle each other
    print
    for i1 in xrange(num_bots):
        bot1 = bots[i1]
        for i2 in xrange(i1+1, num_bots):
            bot2 = bots[i2]

            bot1_name = bot1.name()
            bot2_name = bot2.name()
            print "'%s' vs '%s'" % (bot1_name, bot2_name)
            (bot1_points, bot2_points) = battle(bot1, bot2)
            print "\t%d points for '%s'" % (bot1_points, bot1_name)
            print "\t%d points for '%s'" % (bot2_points, bot2_name)
            print

...
```

take directory of modules from command line

tournament.py

```
...
if __name__ == "__main__":
    assert len(sys.argv) == 2
    bots_dir = sys.argv[1]
    assert os.path.isdir(bots_dir)

    #
    # Load modules contained in bots directory
    bots = load_bots(bots_dir)
    num_bots = len(bots)

    #
    # Bots battle each other
    print
    for i1 in xrange(num_bots):
        bot1 = bots[i1]
        for i2 in xrange(i1+1, num_bots):
            bot2 = bots[i2]

            bot1_name = bot1.name()
            bot2_name = bot2.name()
            print "'%s' vs '%s'" % (bot1_name, bot2_name)
            (bot1_points, bot2_points) = battle(bot1, bot2)
            print "\t%d points for '%s'" % (bot1_points, bot1_name)
            print "\t%d points for '%s'" % (bot2_points, bot2_name)
            print

...
```

take directory of modules from command line

load modules as list of module objects

tournament.py

```
...
if __name__ == "__main__":
    assert len(sys.argv) == 2
    bots_dir = sys.argv[1]
    assert os.path.isdir(bots_dir)

    #
    # Load modules contained in bots directory
    bots = load_bots(bots_dir)
    num_bots = len(bots)

    #
    # Bots battle each other
    print
    for i1 in xrange(num_bots):
        bot1 = bots[i1]
        for i2 in xrange(i1+1, num_bots):
            bot2 = bots[i2]

            bot1_name = bot1.name()
            bot2_name = bot2.name()
            print "'%s' vs '%s'" % (bot1_name, bot2_name)
            (bot1_points, bot2_points) = battle(bot1, bot2)
            print "\t%d points for '%s'" % (bot1_points, bot1_name)
            print "\t%d points for '%s'" % (bot2_points, bot2_name)
            print

...
```

take directory of modules from command line

load modules as list of module objects

***battle* function handles match between two bots**

tournament.py — *battle()*

```
...

def battle(player_a, player_b, num_rounds=1000):
    a_moves = []
    b_moves = []
    a_points = 0
    b_points = 0

    for _ in xrange(num_rounds):

        if random.randint(0, 1):
            a_move = player_a.move(a_moves, b_moves)
            b_move = player_b.move(b_moves, a_moves)
        else:
            b_move = player_b.move(b_moves, a_moves)
            a_move = player_a.move(a_moves, b_moves)

        outcome = beats(a_move, b_move)
        if outcome == "W":
            a_points += WIN_POINTS
            b_points += LOSE_POINTS
        elif outcome == "L":
            a_points += LOSE_POINTS
            b_points += WIN_POINTS
        elif outcome =="D":
            a_points += DRAW_POINTS
            b_points += DRAW_POINTS

        a_moves.append(a_move)
        b_moves.append(b_move)

    return (a_points, b_points)

...
```

tournament.py — *battle()*

```
...
def battle(player_a, player_b, num_rounds=1000):
    a_moves = []
    b_moves = []
    a_points = 0
    b_points = 0

    for _ in xrange(num_rounds):

        if random.randint(0, 1):
            a_move = player_a.move(a_moves, b_moves)
            b_move = player_b.move(b_moves, a_moves)
        else:
            b_move = player_b.move(b_moves, a_moves)
            a_move = player_a.move(a_moves, b_moves)

        outcome = beats(a_move, b_move)
        if outcome == "W":
            a_points += WIN_POINTS
            b_points += LOSE_POINTS
        elif outcome == "L":
            a_points += LOSE_POINTS
            b_points += WIN_POINTS
        elif outcome =="D":
            a_points += DRAW_POINTS
            b_points += DRAW_POINTS

        a_moves.append(a_move)
        b_moves.append(b_move)

    return (a_points, b_points)

...
```

keep record of moves and points scored so far

tournament.py — *battle()*

```
...
def battle(player_a, player_b, num_rounds=1000):
    a_moves = []
    b_moves = []
    a_points = 0
    b_points = 0

    for _ in xrange(num_rounds):

        if random.randint(0, 1):
            a_move = player_a.move(a_moves, b_moves)
            b_move = player_b.move(b_moves, a_moves)
        else:
            b_move = player_b.move(b_moves, a_moves)
            a_move = player_a.move(a_moves, b_moves)

        outcome = beats(a_move, b_move)
        if outcome == "W":
            a_points += WIN_POINTS
            b_points += LOSE_POINTS
        elif outcome == "L":
            a_points += LOSE_POINTS
            b_points += WIN_POINTS
        elif outcome =="D":
            a_points += DRAW_POINTS
            b_points += DRAW_POINTS

        a_moves.append(a_move)
        b_moves.append(b_move)

    return (a_points, b_points)

...
```

keep record of moves and points scored so far

make both players choose their move for this round

(order at random, in case of shenanigans)

tournament.py — *battle()*

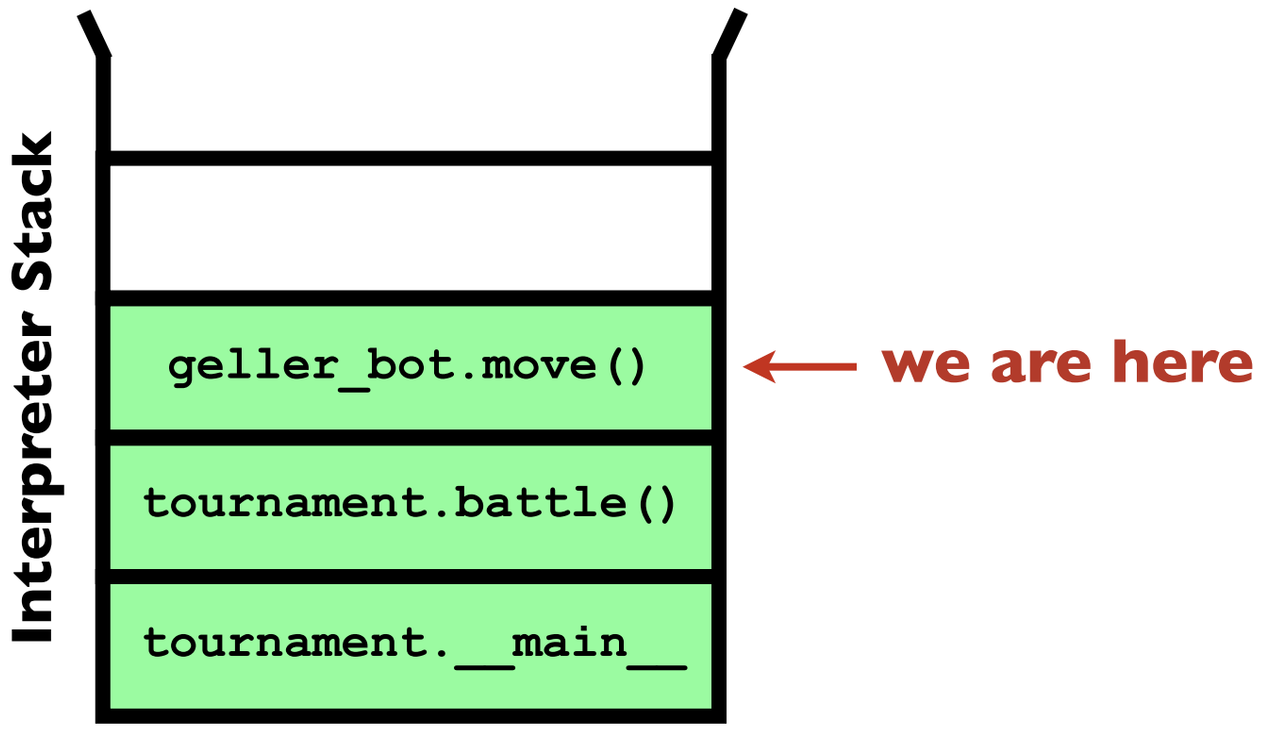
```
...
def battle(player_a, player_b, num_rounds=1000):
    a_moves = []
    b_moves = []
    a_points = 0
    b_points = 0

    for _ in xrange(num_rounds):

        if random.randint(0, 1):
            a_move = player_a.move(a_moves, b_moves)
            b_move = player_b.move(b_moves, a_moves)
        else:
            b_move = player_b.move(b_moves, a_moves)
            a_move = player_a.move(a_moves, b_moves)

        outcome = beats(a_move, b_move)
        if outcome == "W":
            a_points += WIN_POINTS
            b_points += LOSE_POINTS
        elif outcome == "L":
            a_points += LOSE_POINTS
            b_points += WIN_POINTS
        elif outcome =="D":
            a_points += DRAW_POINTS
            b_points += DRAW_POINTS

        a_moves.append(a_move)
        b_moves.append(b_move)

    return (a_points, b_points)

...
```

keep record of moves and points scored so far

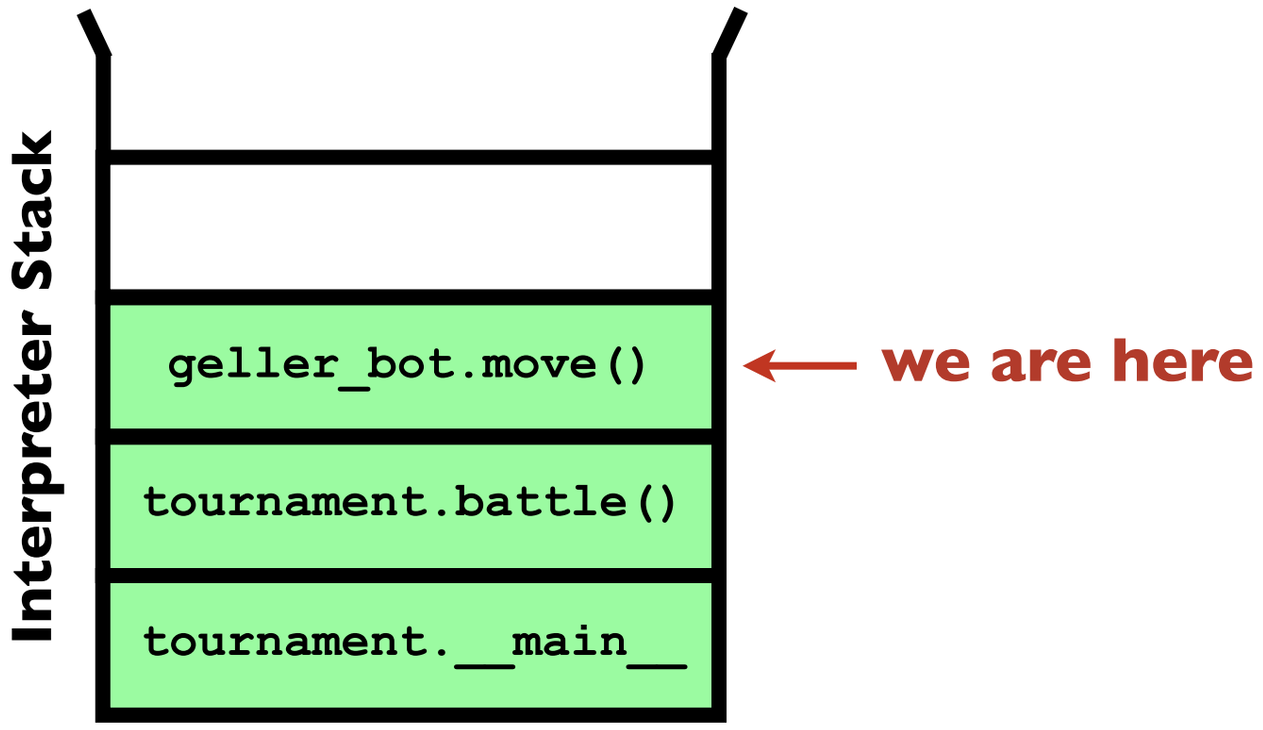
make both players choose their move for this round ***(order at random, in case of shenanigans)***

...and allocate points based on outcome...

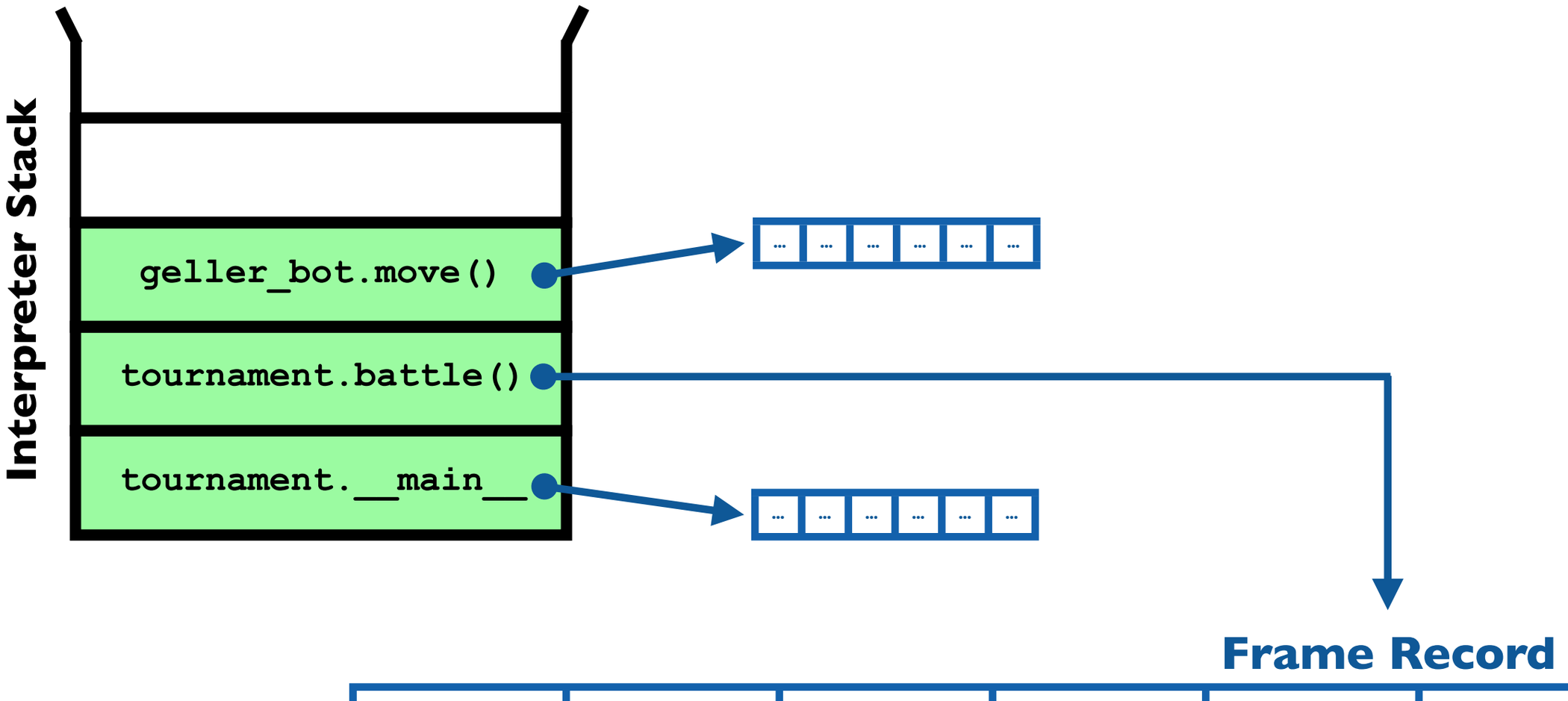
State of the stack when GellerBot's `move` is called...



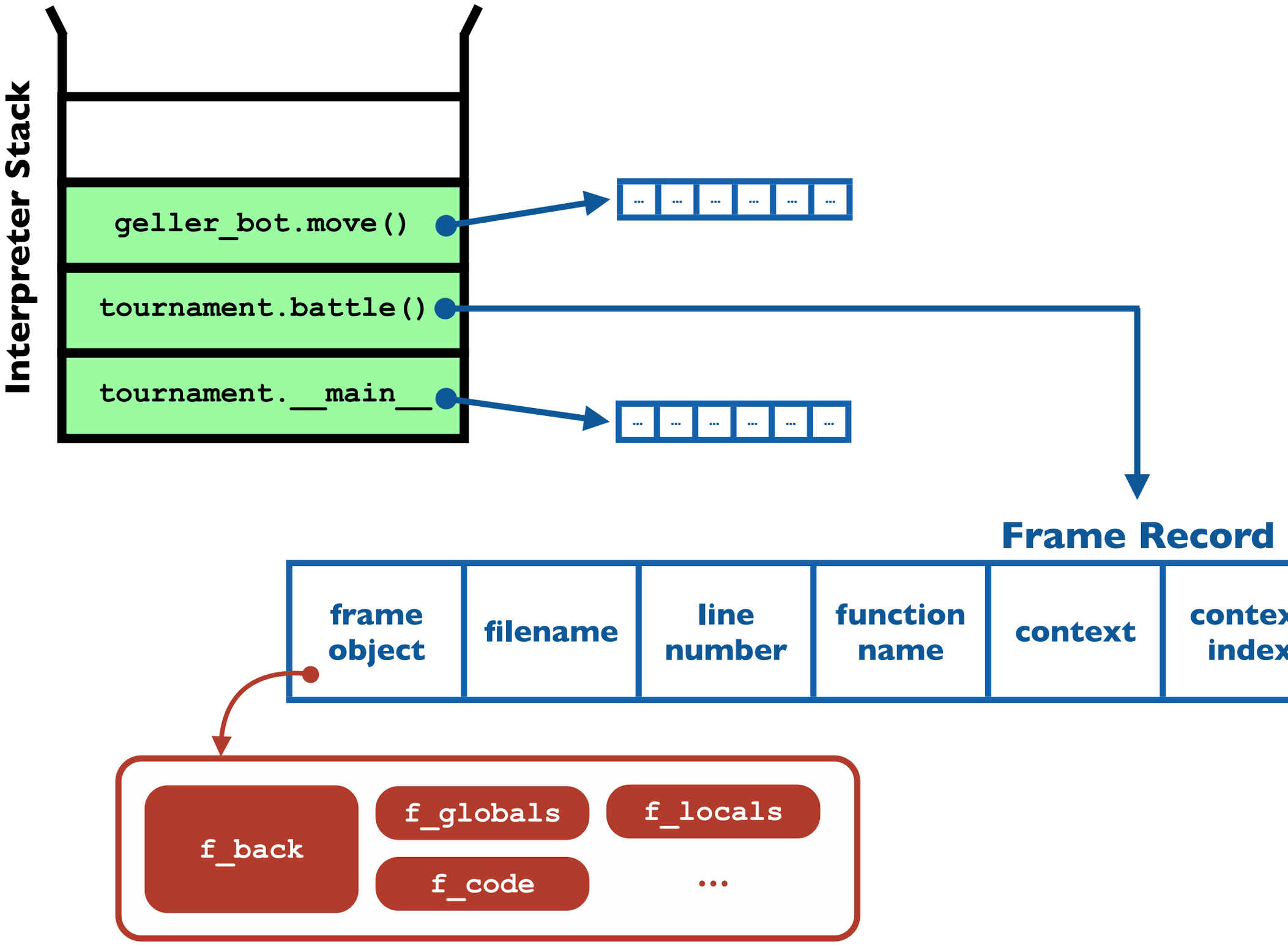
State of the stack when GellerBot's `move` is called...

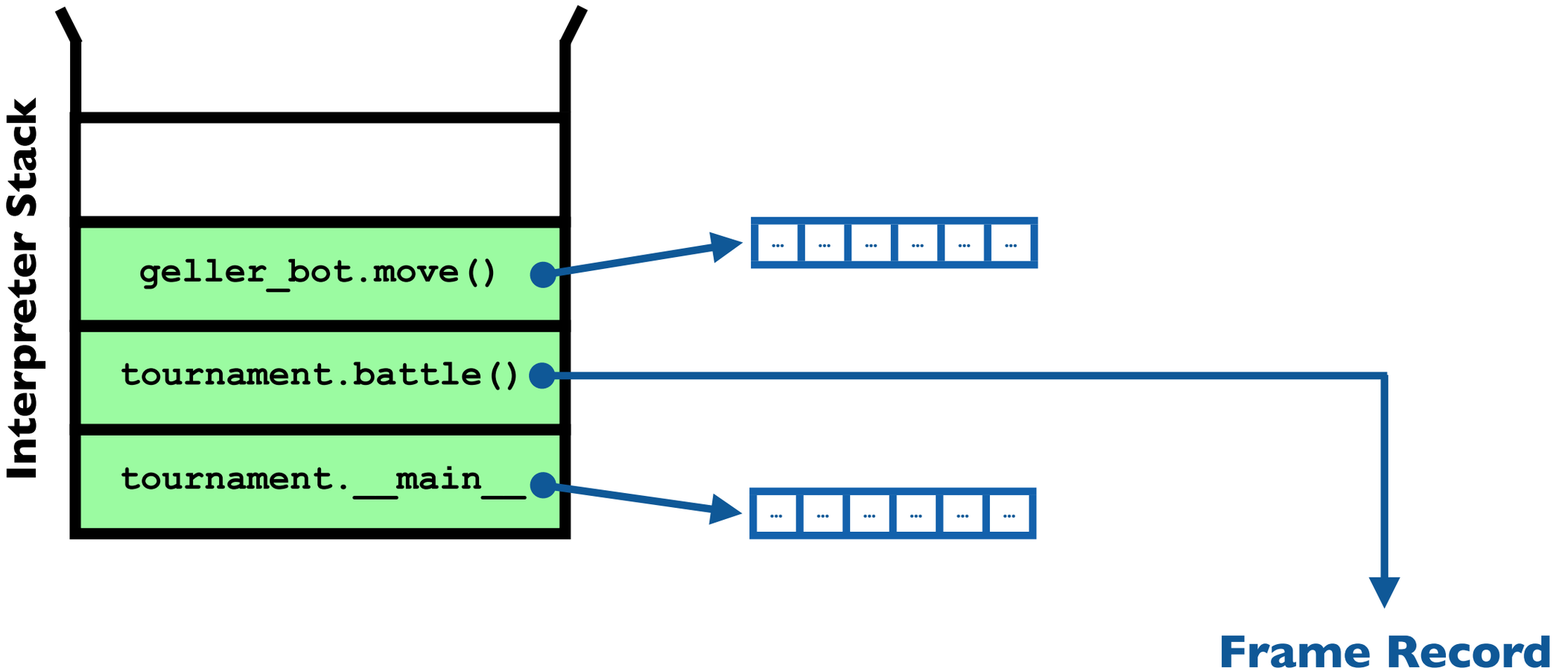


So, let's try and find the opponent's module in this sequence of calls!

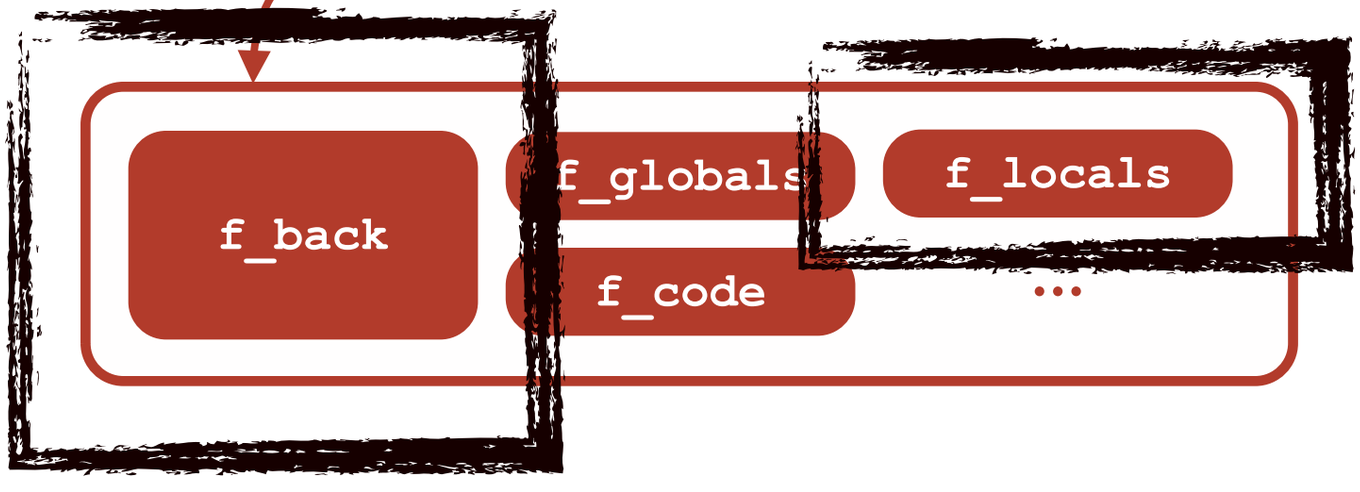


frame object	filename	line number	function name	context	context index


Interpreter Stack
geller_bot.move()
tournament.battle()
tournament.__main__
Frame Record
frame object
filename
line number
function name
context
context index
f_back
f_globals
f_locals
f_code
...



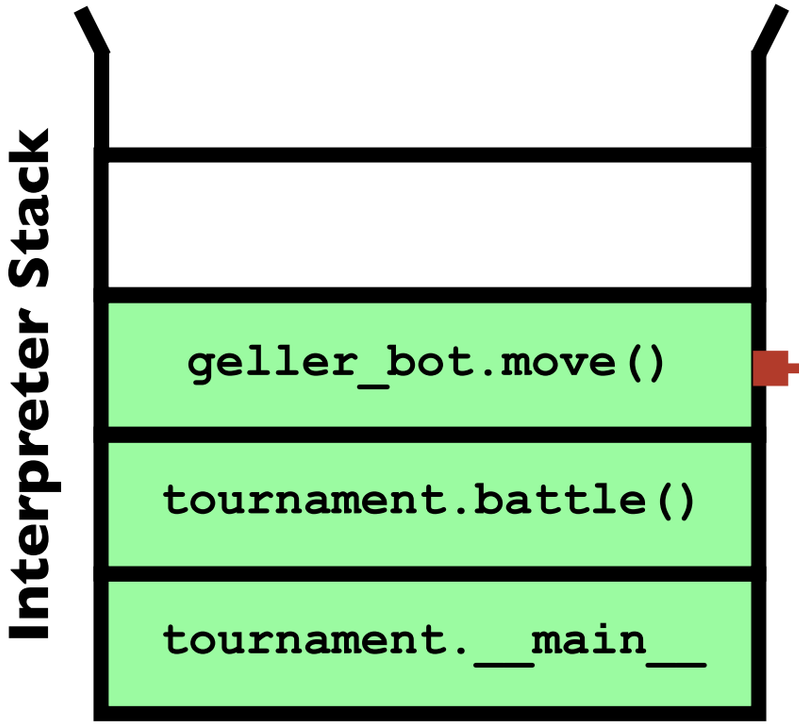
for traversing the stack



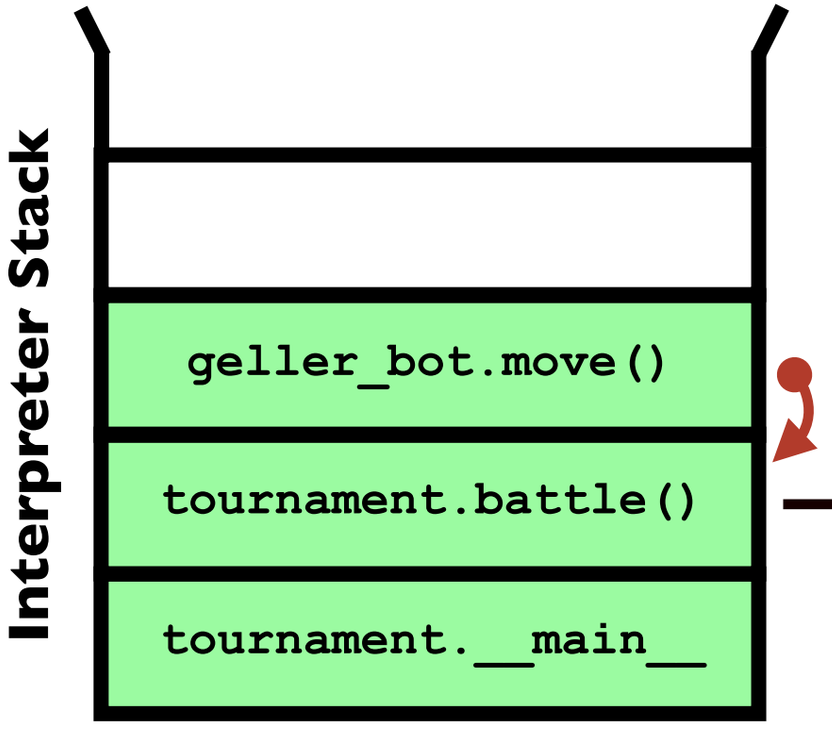
to check if an opponent bot is in scope

inspect.**currentframe**()

Return the frame object for the caller's stack frame.




```
current_frame = inspect.currentframe()

ancestor_frame = current_frame.f_back
while ancestor_frame is not None:

    #
    # ...do something with the ancestor frame...
    #

    # grab next ancestor
    ancestor_frame = ancestor_frame.f_back
```



We've found the *battle()* frame when...

...we see a frame that has at least two 'bot' modules among its local variables (i.e., in the `f_locals` dict)

Module checking...

```
isinstance(var, types.ModuleType)
```

Bot checking...

```
hasattr(var, 'move') and
hasattr(var, 'name')
```


```
def move(my_moves, opp_moves):

    curr_frame = inspect.currentframe()

    #
    # Find the opponent's module
    battle_frame = find_battle_frame(curr_frame)
    if battle_frame is None:
        return random.choice(('R', 'P', 'S'))

    bots = get_local_bot_modules(battle_frame)

    this_bot = sys.modules[__name__]
    bots.remove(this_bot)

    opponent = bots.pop()

    #
    # See what the opponent will throw and beat them
    opp_move = opponent.move(opp_moves, my_moves)

    if opp_move == 'R':
        return 'P'
    elif opp_move == 'P':
        return 'S'
    else:
        return 'R'
```


```python
def move(my_moves, opp_moves):
    curr_frame = inspect.currentframe()

    #
    # Find the opponent's module
    battle_frame = find_battle_frame(curr_frame)
    if battle_frame is None:
        return random.choice(('R', 'P', 'S'))

    bots = get_local_bot_modules(battle_frame)

    this_bot = sys.modules[__name__]
    bots.remove(this_bot)

    opponent = bots.pop()

    #
    # See what the opponent will throw and beat them
    opp_move = opponent.move(opp_moves, my_moves)

    if opp_move == 'R':
        return 'P'
    elif opp_move == 'P':
        return 'S'
    else:
        return 'R'
```

traverse stack to get *battle* frame


```python
def move(my_moves, opp_moves):
    curr_frame = inspect.currentframe()

    #
    # Find the opponent's module
    battle_frame = find_battle_frame(curr_frame)
    if battle_frame is None:
        return random.choice(('R', 'P', 'S'))
```

traverse stack to get *battle* frame

```python
    bots = get_local_bot_modules(battle_frame)

    this_bot = sys.modules[__name__]
    bots.remove(this_bot)

    opponent = bots.pop()
```

get an opponent bot from the *battle* frame

```python
    #
    # See what the opponent will throw and beat them
    opp_move = opponent.move(opp_moves, my_moves)

    if opp_move == 'R':
        return 'P'
    elif opp_move == 'P':
        return 'S'
    else:
        return 'R'
```


```
def move(my_moves, opp_moves):
    curr_frame = inspect.currentframe()

    #
    # Find the opponent's module
    battle_frame = find_battle_frame(curr_frame)
    if battle_frame is None:
        return random.choice(('R', 'P', 'S'))

    bots = get_local_bot_modules(battle_frame)

    this_bot = sys.modules[__name__]
    bots.remove(this_bot)

    opponent = bots.pop()

    #
    # See what the opponent will throw and beat them
    opp_move = opponent.move(opp_moves, my_moves)

    if opp_move == 'R':
        return 'P'
    elif opp_move == 'P':
        return 'S'
    else:
        return 'R'
```

traverse stack to get *battle* frame

get an opponent bot from the *battle* frame

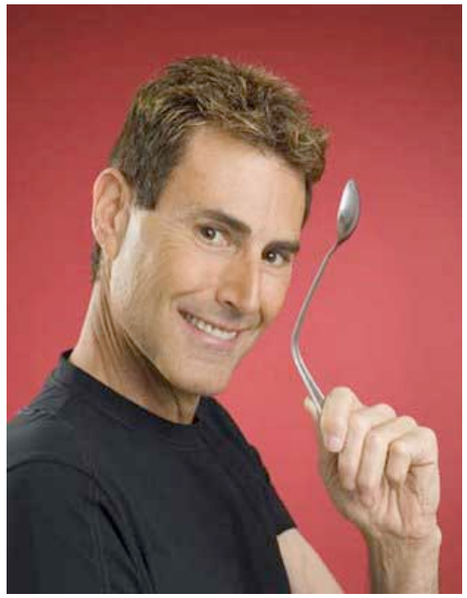
run the opponent's move *(and then crush them!)*


L1

VS

vs

=



```
RPS — bash — 54×13
matt@ouranos: python tournament.py bots_gellerbot1

'Bart Simpson' vs 'Uri Geller'
        1000 points for 'Bart Simpson'
        3000 points for 'Uri Geller'

matt@ouranos:
```


L1

vs

vs

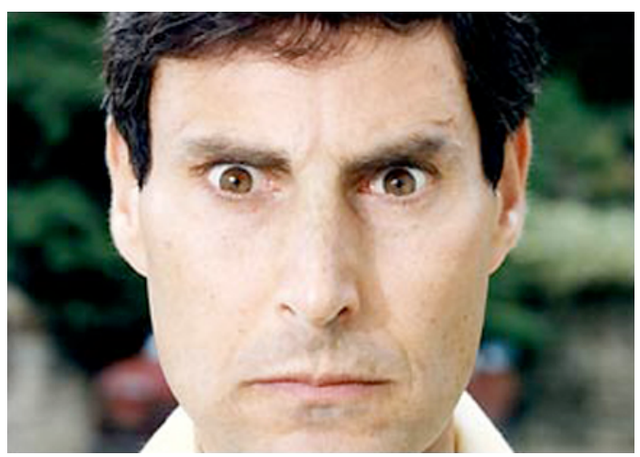
RPS — bash — 60×16

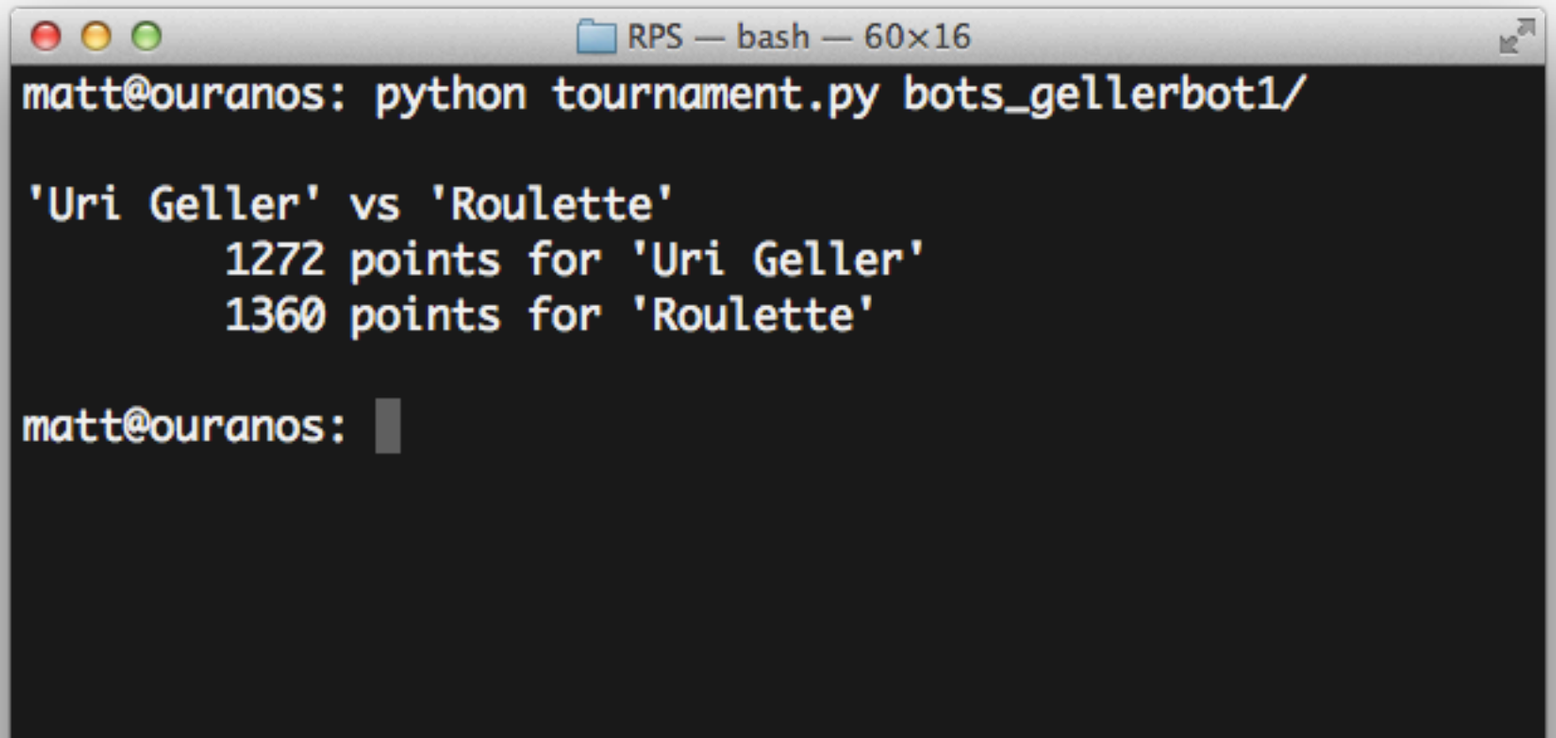
```
matt@ouranos: python tournament.py bots_gellerbot1/

'Uri Geller' vs 'Roulette'
        1272 points for 'Uri Geller'
        1360 points for 'Roulette'

matt@ouranos:
```


L1

vs

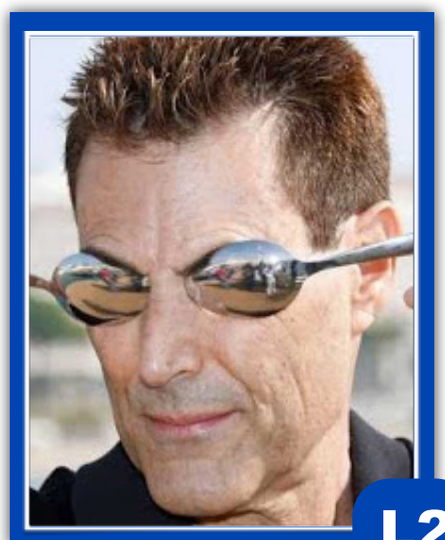
=




RPS — bash — 60×16
matt@ouranos: python tournament.py bots_gellerbot1/
'Uri Geller' vs 'Roulette'
1272 points for 'Uri Geller'
1360 points for 'Roulette'
matt@ouranos:


```
def move(my_moves, opp_moves):
    curr_frame = inspect.currentframe()

    #
    # Find the opponent's module
    battle_frame = find_battle_frame(curr_frame)
    if battle_frame is None:
        return random.choice(('R', 'P', 'S'))

    bots = get_local_bot_modules(battle_frame)

    this_bot = sys.modules[__name__]
    bots.remove(this_bot)

    opponent = bots.pop()

    #
    # See what the opponent will throw and beat them
    rand_state = random.getstate()
    opp_next = opponent.move(opp_moves, my_moves)
    random.setstate(rand_state)

    if opp_move == 'R':
        return 'P'
    elif opp_move == 'P':
        return 'S'
    else:
        return 'R'
```

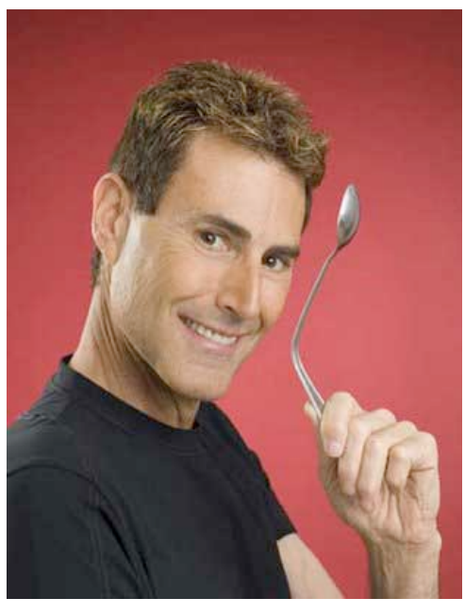
fiddle the *pseudo*random number generator


L2

VS

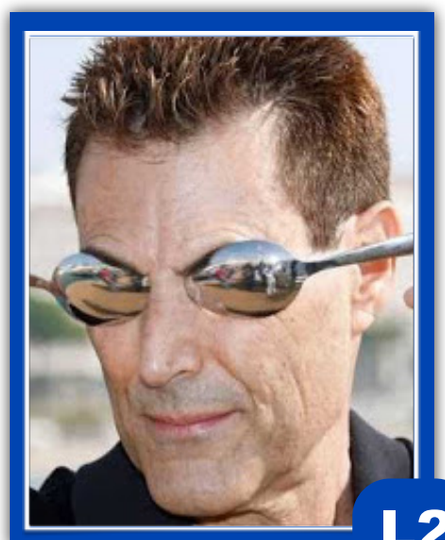
vs

=



RPS — bash — 60×16

```
matt@ouranos: python tournament.py bots_gellerbot2/

'Uri Geller' vs 'Roulette'
        2220 points for 'Uri Geller'
        1180 points for 'Roulette'

matt@ouranos:
```


L2

VS

vs

=

RPS — bash — 76×19

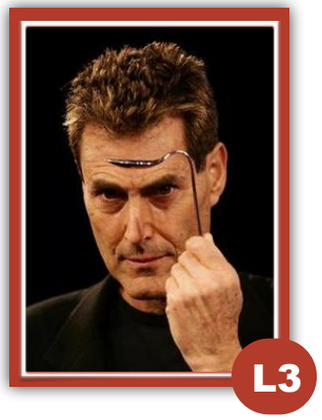
```
    opp_move = opponent.move(opp_moves, my_moves)
  File "bots_gellerbot2/gellerbot_L2.py", line 57, in move
    opp_move = opponent.move(opp_moves, my_moves)
  File "bots_gellerbot2/gellerdoppelbot.py", line 57, in move
    opp_move = opponent.move(opp_moves, my_moves)
  File "bots_gellerbot2/gellerbot_L2.py", line 43, in move
    battle_frame = find_battle_frame(curr_frame)
  File "bots_gellerbot2/gellerbot_L2.py", line 30, in find_battle_frame
    bots = get_local_bot_modules(ancestor_frame)
  File "bots_gellerbot2/gellerbot_L2.py", line 15, in get_local_bot_modules
    if isinstance(val, types.ModuleType):
RuntimeError: maximum recursion depth exceeded while calling a Python object
matt@ouranos:
matt@ouranos:
matt@ouranos:
matt@ouranos:
```

vs

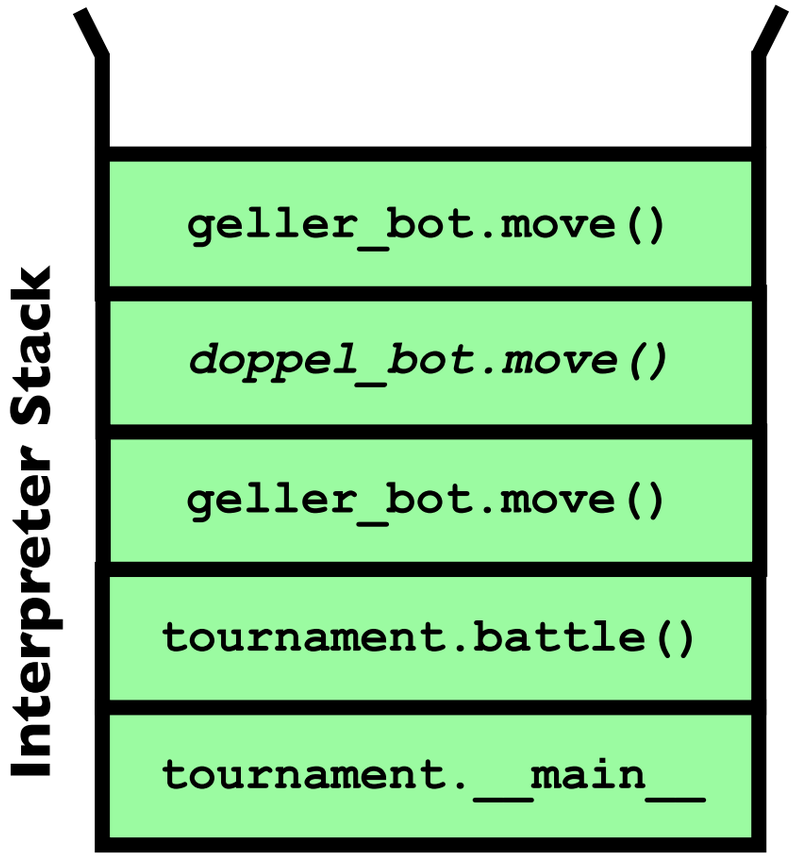
=

RPS — bash — 76×19

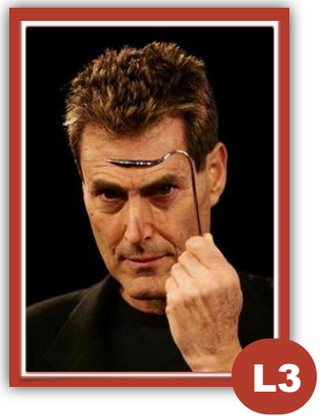
```
    opp_move = opponent.move(opp_moves, my_moves)
  File "bots_gellerbot2/gellerbot_L2.py", line 57, in move
    opp_move = opponent.move(opp_moves, my_moves)
  File "bots_gellerbot2/gellerdoppelbot.py", line 57, in move
    opp_move = opponent.move(opp_moves, my_moves)
  File "bots_gellerbot2/gellerbot_L2.py", line 43, in move
    battle_frame = find_battle_frame(curr_frame)
  File "bots_gellerbot2/gellerbot_L2.py", line 30, in find_battle_frame
    bots = get_local_bot_modules(ancestor_frame)
  File "bots_gellerbot2/gellerbot_L2.py", line 15, in get_local_bot_modules
    if isinstance(val, types.ModuleType):
RuntimeError: maximum recursion depth exceeded while calling a Python object
matt@ouranos:
matt@ouranos:
matt@ouranos:
matt@ouranos:
```



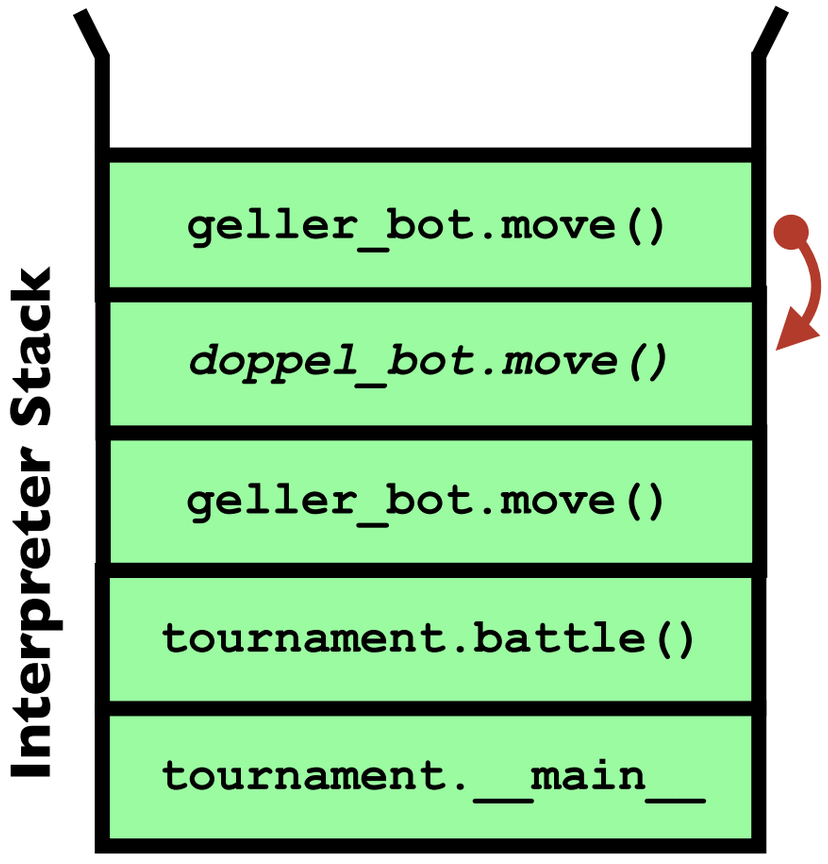
Detecting recursion loops...



Is our `move` getting called called by a bot?

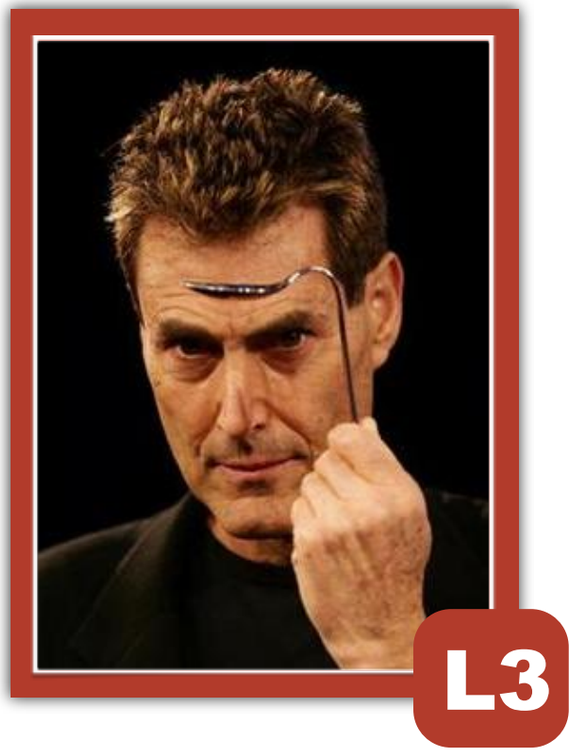


Detecting recursion loops...

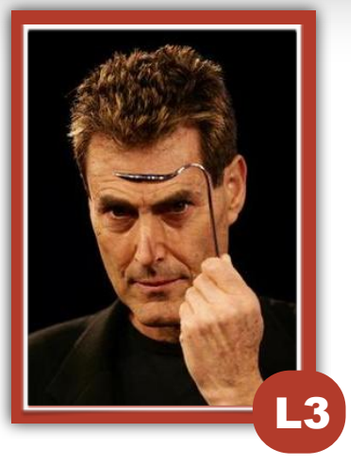


Is our `move` getting called called by a bot?

If we see another `move` frame in the stack, then let's give up and return a random R/P/S




```
RPS — bash — 52×15

matt@ouranos: python tournament.py bots_gellerbot3/

'Uri Geller' vs 'Bart Simpson'
        3000 points for 'Uri Geller'
        1000 points for 'Bart Simpson'

'Uri Geller' vs 'Uri Geller'
        0 points for 'Uri Geller'
        0 points for 'Uri Geller'

'Uri Geller' vs 'Roulette'
        2223 points for 'Uri Geller'
        1149 points for 'Roulette'
```



Manipulating code

A quick glance at Python ASTs

Abstract syntax trees

- A tree representation of Python syntax
- Built-in Python tools:

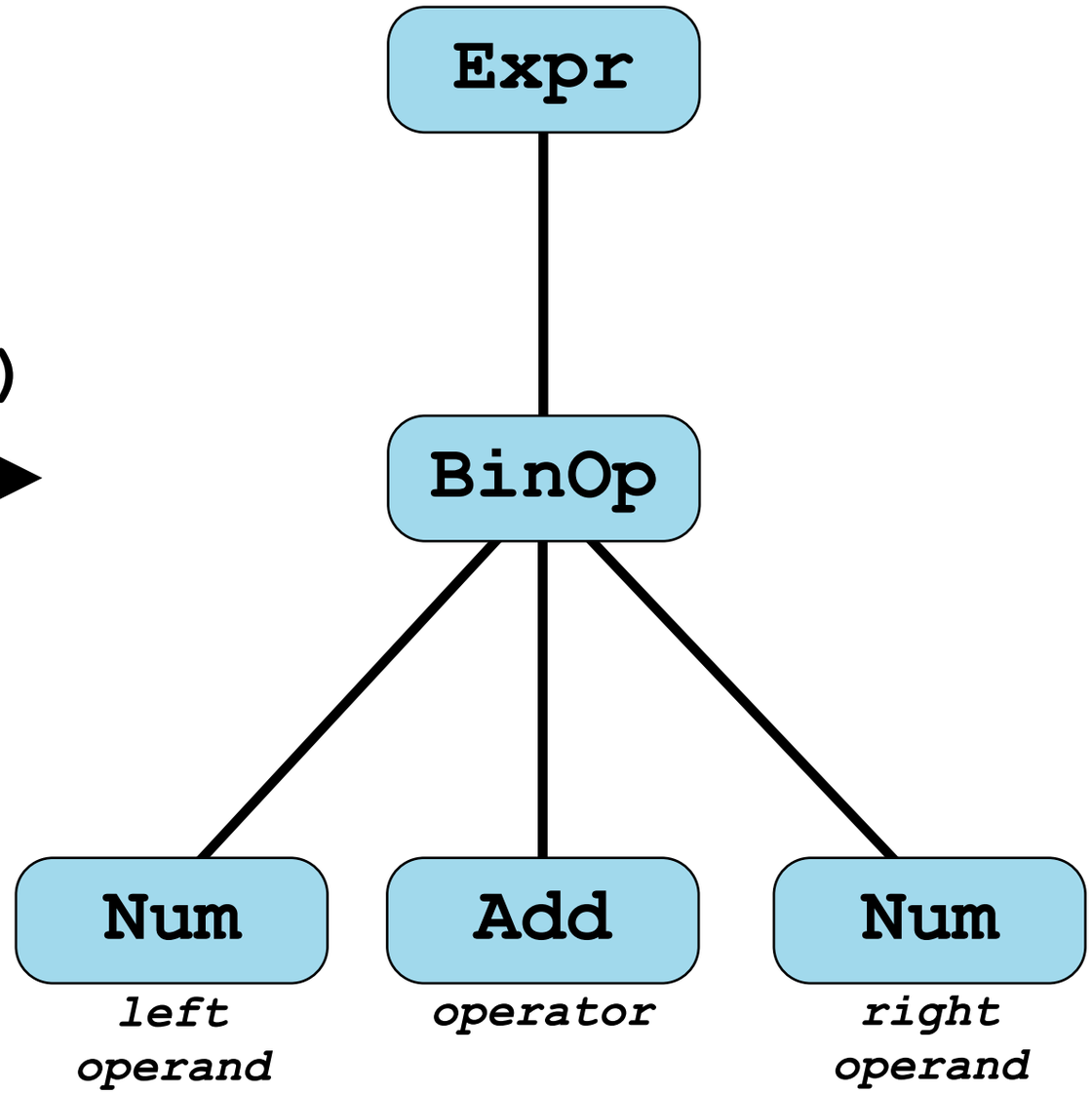
  `ast` – module for ASTs and parsing source code

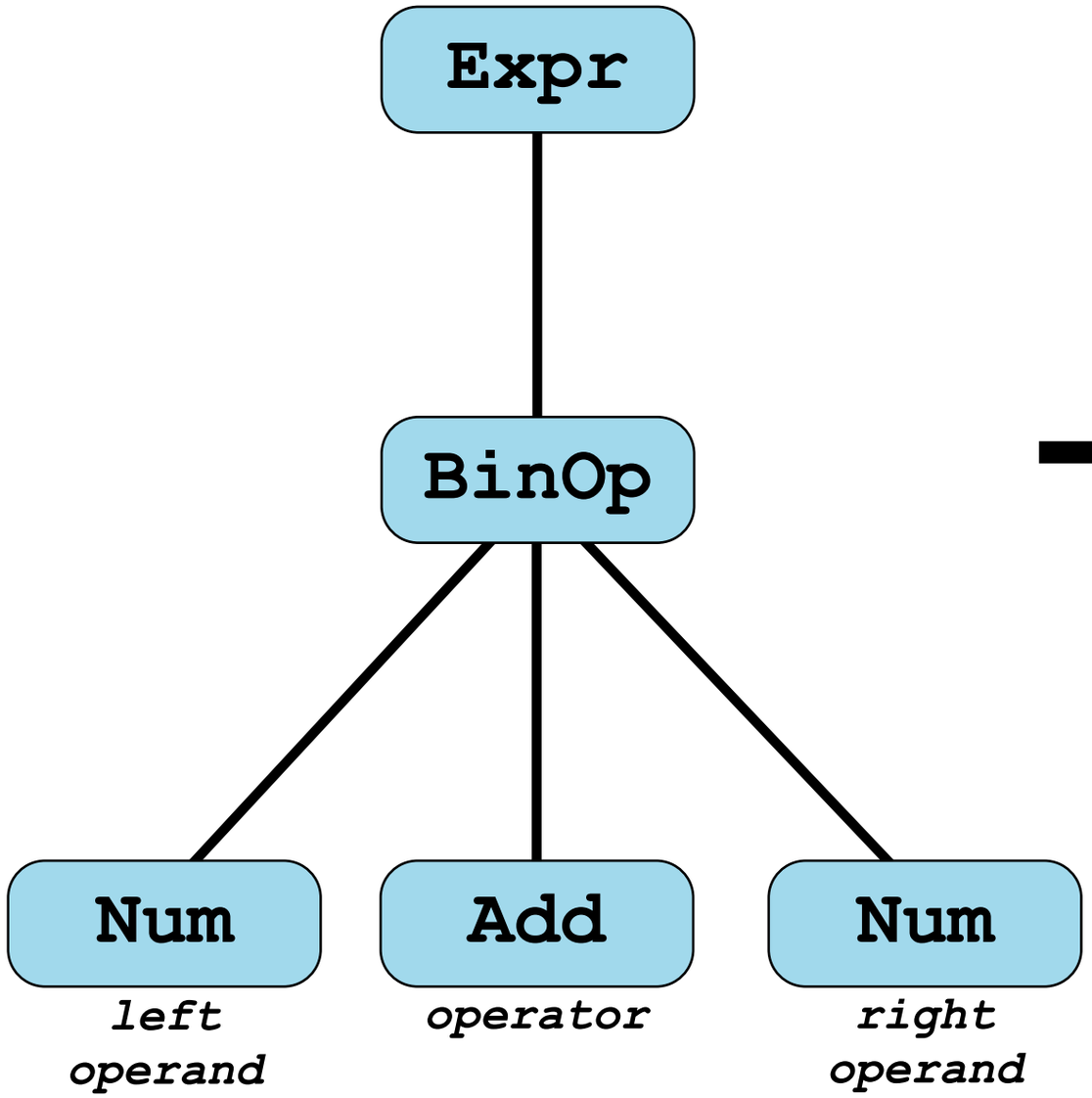
  `compile` – compile an AST to a Python bytecode object *(can also compile from source code string)*

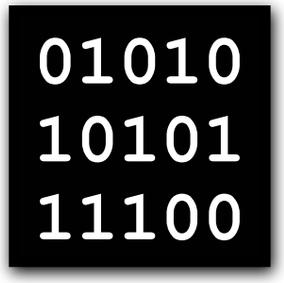
  `exec` – execute a bytecode object

```
3 + 4
```

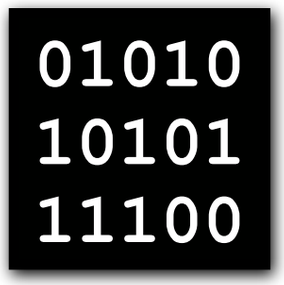
ast.parse()







Python bytecode object

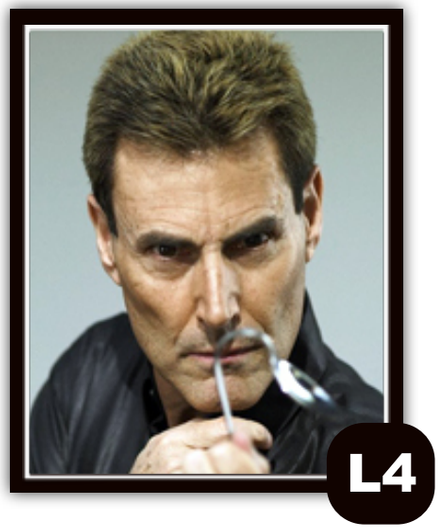


Python bytecode object

exec

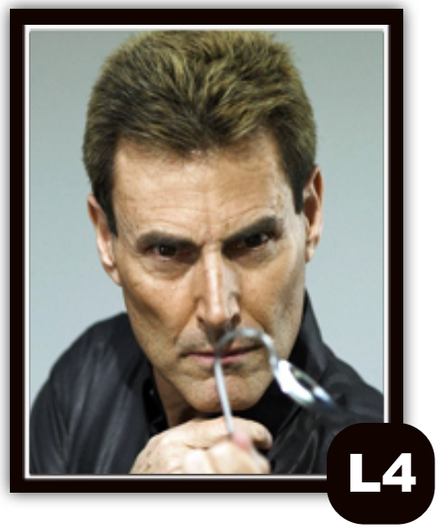
Err...
No actual output, but it did evaluate *3+4*

GellerBot Level 4



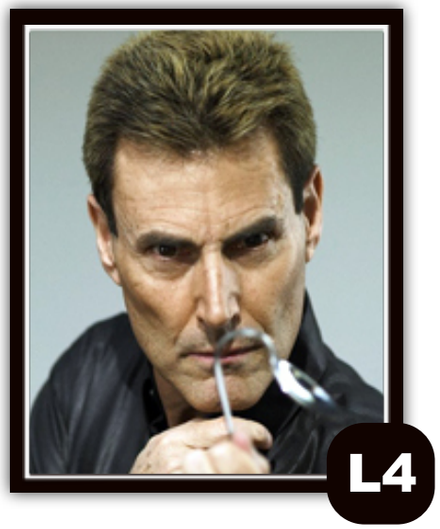
- Locate opponent's `move` function
- Get `move`'s source code
- Parse to AST
- Insert `return 'R'` as first expression
- Replace `move`'s code object with the compiled bytecode
- Beat opponent, *every time*

GellerBot Level 4



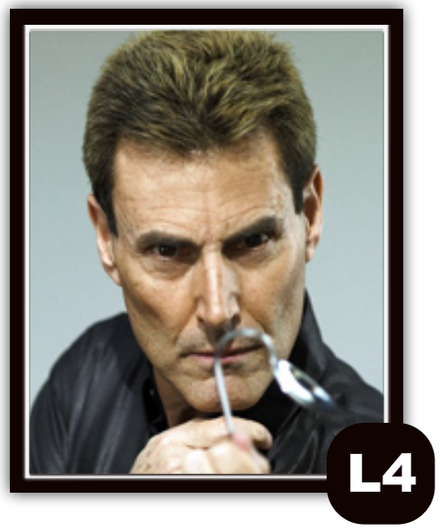
- Locate opponent's `move` function **as before**
- Get `move`'s source code
- Parse to AST
- Insert `return 'R'` as first expression
- Replace `move`'s code object with the compiled bytecode
- Beat opponent, *every time*

GellerBot Level 4



- Locate opponent's `move` function
- Get `move`'s source code — **inspect.getsource()**
- Parse to AST
- Insert `return 'R'` as first expression
- Replace `move`'s code object with the compiled bytecode
- Beat opponent, *every time*

GellerBot Level 4

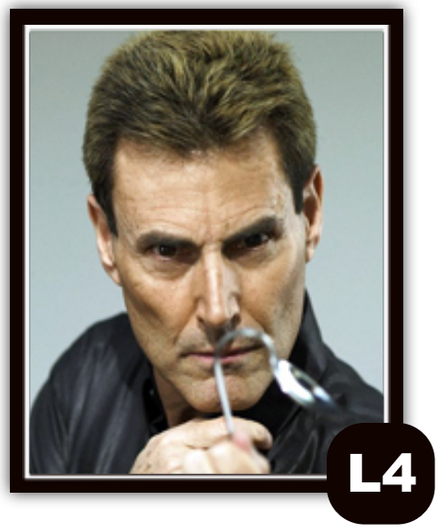


- Locate opponent's `move` function
- Get `move`'s source code
- Parse to AST

ast.parse()

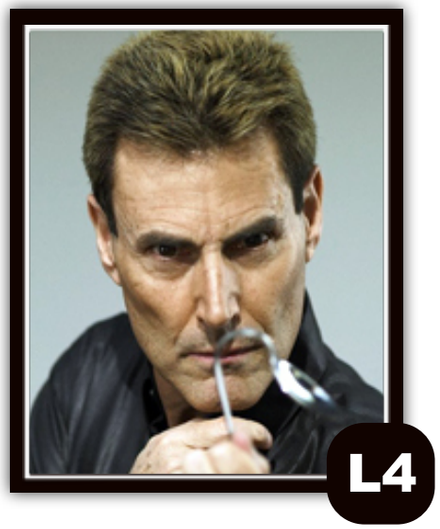
- Insert `return 'R'` as first expression
- Replace `move`'s code object with the compiled bytecode
- Beat opponent, *every time*

GellerBot Level 4



- Locate opponent's `move` function
- Get `move`'s source code
- Parse to AST
- Insert `return 'R'` as first expression
- Replace `move`'s code object with the compiled bytecode
- Beat opponent, *every time*

GellerBot Level 4

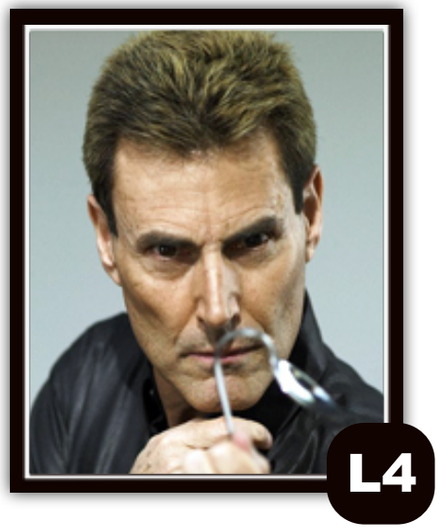


- Locate opponent's `move` function
- Get `move`'s source code
- Parse to AST
- Insert `return 'R'` as first expression
- Replace `move`'s code object with the compiled bytecode

`move.__code__ = ...`

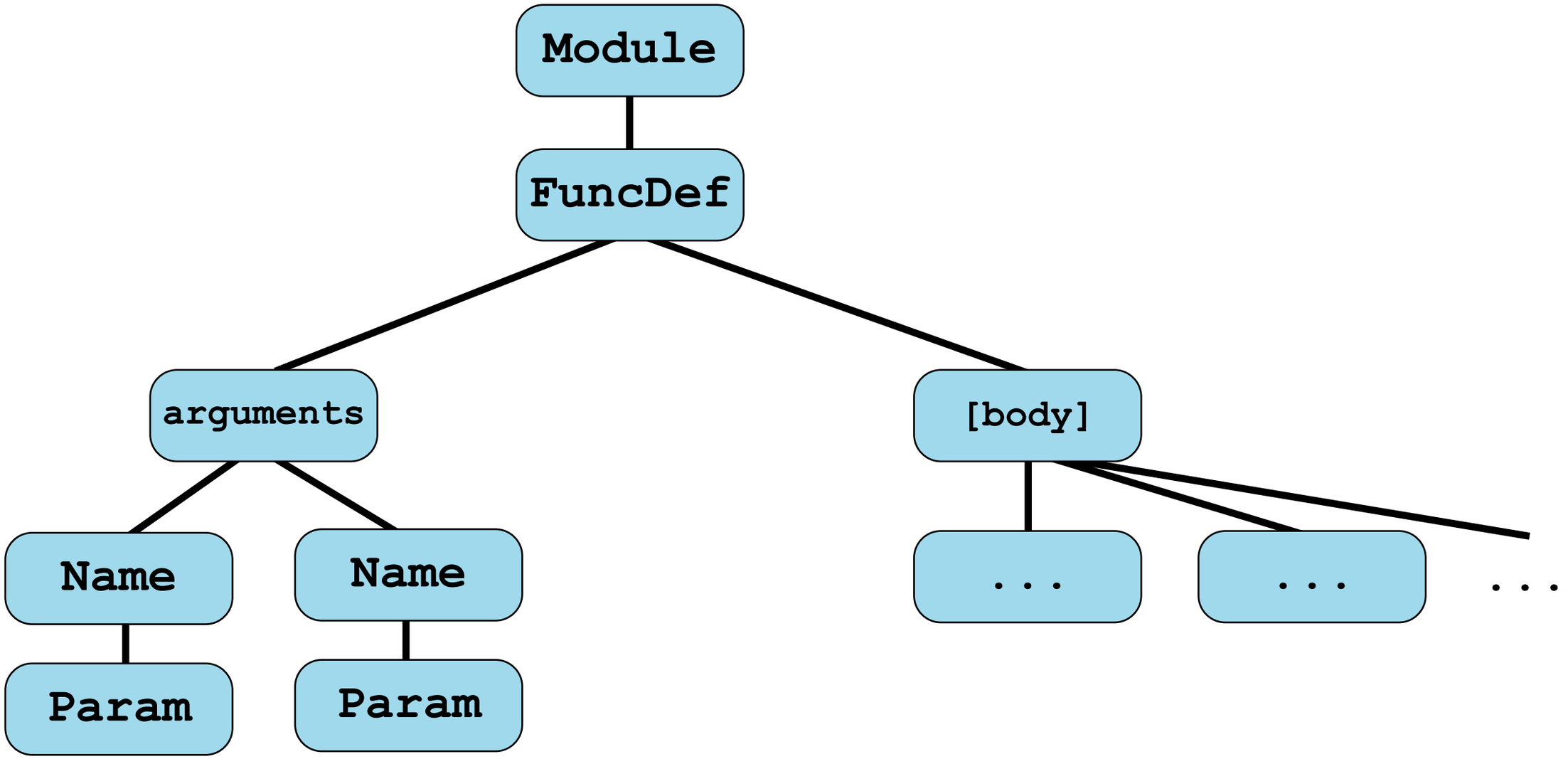
- Beat opponent, *every time*

GellerBot Level 4

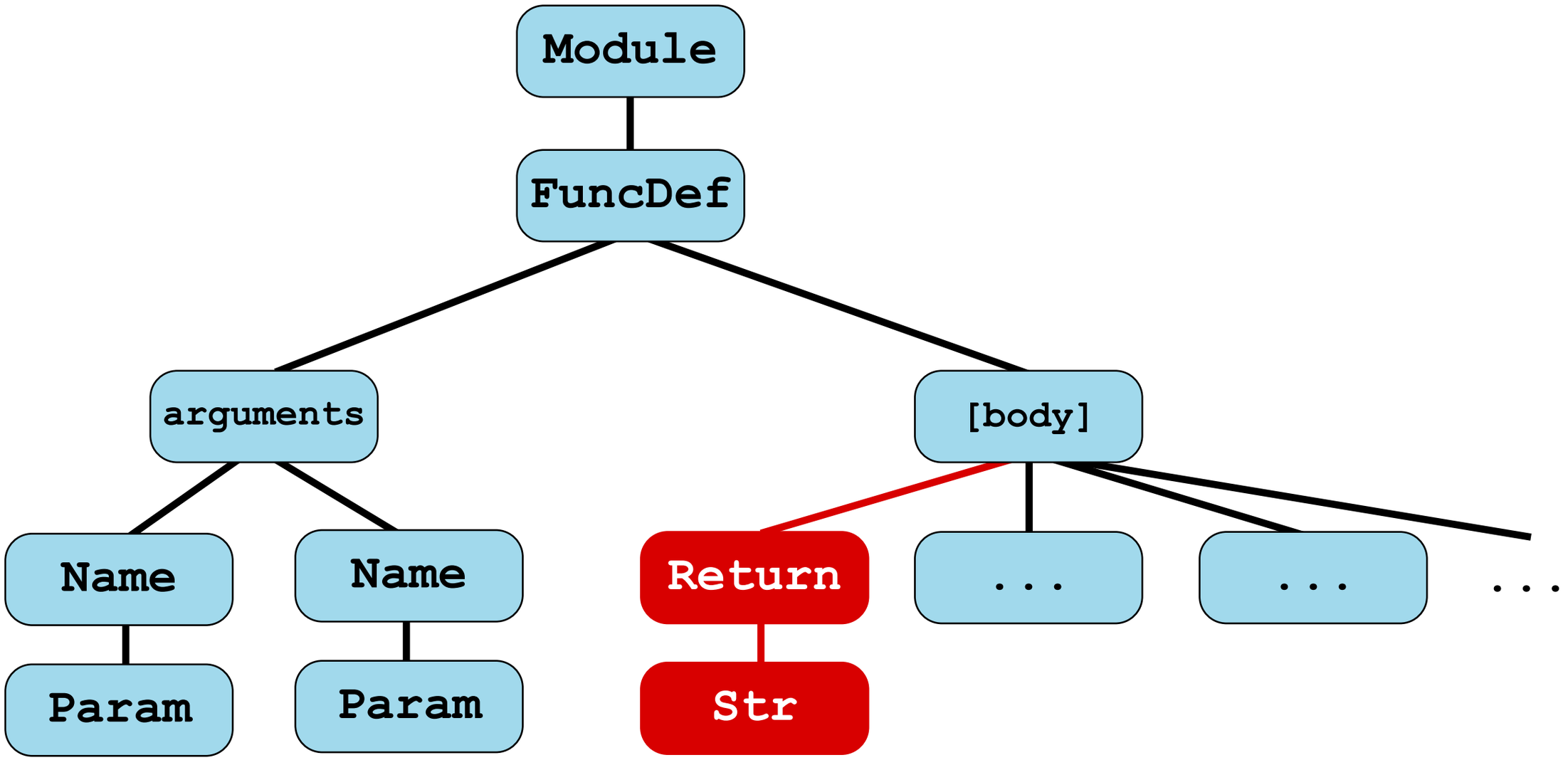


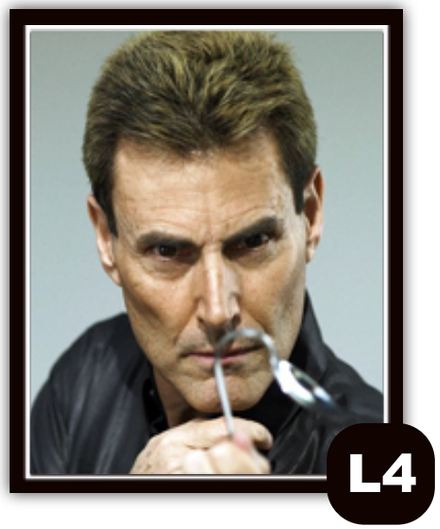
- Locate opponent's `move` function
- Get `move`'s source code
- Parse to AST
- Insert `return 'R'` as first expression
- Replace `move`'s code object with the compiled bytecode
- Beat opponent, *every time* **:-D**

AST for `move`...

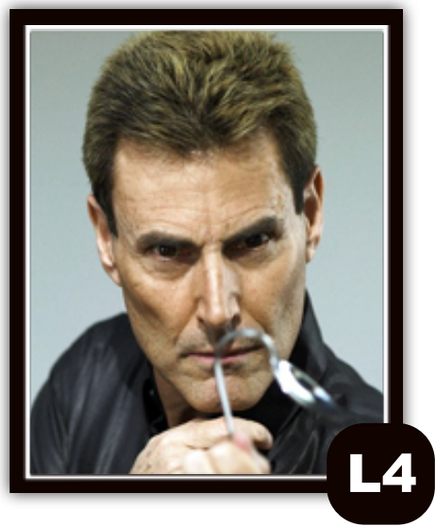


Inserting `return 'R'`…



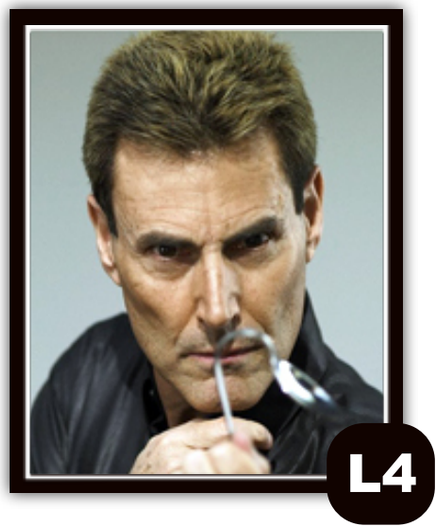


So, how does *GellerBotL4* do?



RPS — bash — 64×17

```
matt@ouranos: python tournament_uri_only.py bots_gellerbot

'Uri Geller' vs 'Bart Simpson'
        3000 points for 'Uri Geller'
        1000 points for 'Bart Simpson'

'Uri Geller' vs 'Uri Geller'
        3000 points for 'Uri Geller'
        1000 points for 'Uri Geller'

'Uri Geller' vs 'Roulette'
        3000 points for 'Uri Geller'
        1000 points for 'Roulette'

matt@ouranos:
```



RPS — bash — 64×17

```
matt@ouranos: python tournament_uri_only.py bots_gellerbot

'Uri Geller' vs 'Bart Simpson'
        3000 points for 'Uri Geller'
        1000 points for 'Bart Simpson'

'Uri Geller' vs 'Uri Geller'
        3000 points for 'Uri Geller'*
        1000 points for 'Uri Geller'

'Uri Geller' vs 'Roulette'
        3000 points for 'Uri Geller'
        1000 points for 'Roulette'

matt@ouranos:
```

*** assuming ‘*our*’ *GellerBot* goes first**

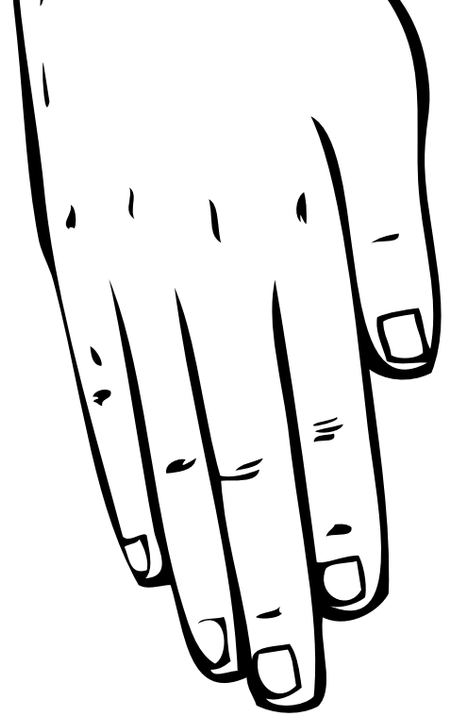
Python and meta-programming

- Many popular **useful** features: dir(), help(), decorators, descriptors, metaclasses, & more
- But also great support for more '*esoteric*' uses...
  - Handling code as abstract syntax trees
  - Inspecting runtime objects
  - Viewing the interpreter stack
  - Compilation to bytecode at runtime

@voxmjw

http://mattjw.net

mattjw@mattjw.net



Code for this talk available on GitHub...

/mattjw/rps_metaprogramming

Slides and photo attribution online at...

http://www.mattjw.net/2014/02/rps-metaprogramming/

Thanks for listening!

Any questions?

Rock, paper, scissors art:
http://www.clker.com/clipart-rock-scissors-paper-1.html

Escher - Drawing Hands
http://alrartblog.blogspot.co.uk/2012/10/mcescher.html

Social media icons:
http://www.pixelfrau.com/free-black-circle-social-media-icons/

Roulette:
http://www.silveroakcasino.com/blog/online-roulette/martingale-roulette-system.html

Uri Gellers:
http://www.mindfulurigeller.com/uri-geller-mindful-mind-reading-club-music-psychic-powers
http://rote-hahn.blogspot.co.uk/2012/10/uri-geller.html
http://latcrossword.blogspot.co.uk/2010/09/f-r-i-d-y-17-2010-john-lampkin.html
http://exequy.wordpress.com/2010/12/06/uri-geller/
http://www.occultopedia.com/g/geller.htm
http://beforeitsnews.com/alternative/2013/06/uri-geller-psychic-spy-the-spoon-benders-secret-life-revealed-2687008.html