Project 1: The Game of Hog

`Project`地址：Project 1: The Game of Hog

Rules

有两个玩家，轮流进行回合，玩家选择骰子的数量（最多10个），得到每个骰子的点数，将点数之和加入总分中，总分先达到`100`即获胜。

`Sow Sad`：如果有一个骰子点数为`1`，本回合得分为`1`

`Piggy Points`：如果选择`0`个骰子，本回合获得`k + 3`分，`k`为对手分数的平方的位数中值最小的一位。

`More Boar`：回合结束后，如果玩家的最高位小于对手的最高位，玩家的次高位小于对手的次高位，继续进行一回合。

若对规则有疑问，请仔细阅读项目网站。

Final Product

最终效果可以参考：Final Product

文件结构

项目目录如下：

```
.
├── calc.py
├── default_graphics.py
├── dice.py
├── gui_files
├── hog_gui.py
├── hog.py
├── ok
├── proj01.ok
├── __pycache__
├── tests
└── ucb.py
```

其中比较重要的有：

- `hog.py`：项目的初始代码
- `dice.py`：关于骰子的函数
- `hog_gui.py`：项目的用户图形化界面（`graphical user interface`）
- `ucb.py`：本课程的实用函数
- `gui_files`：Web GUI 使用的各种内容的目录

执行如下代码可以运行项目：

```
python3 hog_gui.py
```

Phase 1: Simulator

Problem 0 (0 pt)

`dice.py`中定义了两种骰子函数：

- `make_fair_dice(sides)`：`sides`表示骰子的面数，该函数返回一个函数`dice`，`dice`会随机返回`[a, b]`中的一个值。
- `make_test_dice(*outcomes)`：`outcomes`是一个正整数序列，该函数返回一个函数`dice`，`dice`会顺序、周期性地返回`outcomes`中的值，该骰子函数主要用作测试。

运行如下代码进行解锁测评：

```
python3 ok -q 00 -u --local
```

结果如下（第二题读错题了，要的是“摇”一个6个面的骰子的方法）：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 0 > Suite 1 > Case 1
(cases remaining: 2)

>>> from hog import *
>>> test_dice = make_test_dice(4, 1, 2)
>>> test_dice()
? 4
-- OK! --

>>> test_dice() # Second call
? 1
-- OK! --

>>> test_dice() # Third call
? 2
-- OK! --

>>> test_dice() # Fourth call
? 4
-- OK! --

---------------------------------------------------------------------
Question 0 > Suite 2 > Case 1
(cases remaining: 1)
```

```
Q: Which of the following is the correct way to "roll" a fair, six-sided die?
Choose the number of the correct choice:
0) six_sided()
1) make_fair_dice(6)
2) make_test_dice(6)
3) six_sided
? 1
-- Not quite. Try again! --

Choose the number of the correct choice:
0) six_sided()
1) make_fair_dice(6)
2) make_test_dice(6)
3) six_sided
? 0
-- OK! --


---------------------------------------------------------------------
OK! All cases for Question 0 unlocked.

Cannot backup when running ok with --local.
```

Problem 1 (2 pt)

实现`roll_dice(num_rolls, dice=six_sided)`函数，`num_rolls`为骰子数量，`dice`为骰子函数，返回该回合的得分，需要考虑`Sow Sad`规则。

`注意`：哪怕中途有骰子出现了`1`，也要摇完`num_rolls`次骰子，否则测评会出错。

运行如下代码进行解锁测评：

```
python3 ok -q 01 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 1 > Suite 1 > Case 1
(cases remaining: 59)

>>> from hog import *
>>> roll_dice(2, make_test_dice(4, 6, 1))
? 10
-- OK! --
```

```
---------------------------------------------------------------------
Question 1 > Suite 1 > Case 2
(cases remaining: 58)

>>> from hog import *
>>> roll_dice(3, make_test_dice(4, 6, 1))
? 11
-- Not quite. Try again! --

? 1
-- OK! --

---------------------------------------------------------------------
Question 1 > Suite 1 > Case 3
(cases remaining: 57)

>>> from hog import *
>>> roll_dice(4, make_test_dice(2, 2, 3))
? 9
-- OK! --

---------------------------------------------------------------------
Question 1 > Suite 1 > Case 4
(cases remaining: 56)

>>> from hog import *
>>> a = roll_dice(4, make_test_dice(1, 2, 3))
>>> a # check that the value is being returned, not printed
? 1
-- OK! --

---------------------------------------------------------------------
Question 1 > Suite 1 > Case 5
(cases remaining: 55)

>>> from hog import *
>>> counted_dice = make_test_dice(4, 1, 2, 6)
>>> roll_dice(3, counted_dice)
? 1
-- OK! --

>>> # Make sure you call dice exactly num_rolls times!
>>> # If you call it fewer or more than that, it won't be at the right spot in
the cycle for the next roll
>>> # Note that a return statement within a loop ends the loop
>>> roll_dice(1, counted_dice)
? 6
-- OK! --

---------------------------------------------------------------------
Question 1 > Suite 1 > Case 6
(cases remaining: 54)

>>> from hog import *
>>> roll_dice(9, make_test_dice(6))
```

```
? 54
-- OK! --

>>> roll_dice(7, make_test_dice(2, 2, 2, 2, 2, 2, 1))
? 1
-- OK! --

---------------------------------------------------------------------
OK! All cases for Question 1 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下：

```
def roll_dice(num_rolls, dice=six_sided):
    """Simulate rolling the DICE exactly NUM_ROLLS > 0 times. Return the sum of
    the outcomes unless any of the outcomes is 1. In that case, return 1.

    num_rolls:  The number of dice rolls that will be made.
    dice:       A function that simulates a single dice roll outcome.
    """
    # These assert statements ensure that num_rolls is a positive integer.
    assert type(num_rolls) == int, 'num_rolls must be an integer.'
    assert num_rolls > 0, 'Must roll at least once.'
    # BEGIN PROBLEM 1
    "*** YOUR CODE HERE ***"
    sum = 0
    pt_one = False
    for i in range(0, num_rolls):
        pt = dice()
        if pt == 1:
            pt_one = True
        sum += pt
    return (1 if pt_one else sum)
    # END PROBLEM 1
```

可以使用如下命令进行测评：

```
python3 ok -q 01 --local
```

Problem 2 (1 pt)

实现`piggy_points(score)`函数，`score`是对手的分数，`Piggy Points`规则见上，返回本回合的得分。

运行如下代码进行解锁测评：

```
python3 ok -q 02 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
```

```
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 2 > Suite 1 > Case 1
(cases remaining: 36)

>>> from hog import *
>>> import tests.construct_check as test
>>> piggy_points(4)
? 4
-- OK! --

---------------------------------------------------------------------
Question 2 > Suite 1 > Case 2
(cases remaining: 35)

>>> from hog import *
>>> import tests.construct_check as test
>>> piggy_points(10)
? 3
-- OK! --

---------------------------------------------------------------------
Question 2 > Suite 1 > Case 3
(cases remaining: 34)

>>> from hog import *
>>> import tests.construct_check as test
>>> piggy_points(94)
? 6
-- OK! --

---------------------------------------------------------------------
Question 2 > Suite 1 > Case 4
(cases remaining: 33)

>>> from hog import *
>>> import tests.construct_check as test
>>> piggy_points(0)
? 3
-- OK! --

---------------------------------------------------------------------
Question 2 > Suite 1 > Case 5
(cases remaining: 32)

>>> from hog import *
>>> import tests.construct_check as test
```

```
>>> a = piggy_points(24)
>>> a # check that the value is being returned, not printed
? 8
-- OK! --

---------------------------------------------------------------------
OK! All cases for Question 2 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下：

```
def piggy_points(score):
    """Return the points scored from rolling 0 dice.

    score:  The opponent's current score.
    """
    # BEGIN PROBLEM 2
    "*** YOUR CODE HERE ***"
    def smallest_digit(num):
        if num < 10:
            return num
        else:
            return min(num % 10, smallest_digit(num // 10))

    return smallest_digit(score * score) + 3
    # END PROBLEM 2
```

可以使用如下命令进行测评：

```
python3 ok -q 02 --local
```

Problem 3 (2 pt)

实现`take_turn(num_rolls, opponent_score, dice=six_sided, goal=GOAL_SCORE)`函数，返回玩家在本回合的得分，参数如下：

- `num_rolls`：骰子的数量
- `opponent_score`：对手的分数
- `dice`：骰子函数
- `goal`：最高分

运行如下代码进行解锁测评：

```
python3 ok -q 03 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 3 > Suite 1 > Case 1
(cases remaining: 10)

>>> from hog import *
>>> take_turn(2, 0, make_test_dice(4, 5, 1))
? 9
-- OK! --

---------------------------------------------------------------------
Question 3 > Suite 1 > Case 2
(cases remaining: 9)

>>> from hog import *
>>> take_turn(3, 0, make_test_dice(4, 6, 1))
? 1
-- OK! --

---------------------------------------------------------------------
Question 3 > Suite 1 > Case 3
(cases remaining: 8)

>>> from hog import *
>>> take_turn(0, 2)
? 7
-- OK! --

---------------------------------------------------------------------
Question 3 > Suite 1 > Case 4
(cases remaining: 7)

>>> from hog import *
>>> take_turn(0, 0)
? 3
-- OK! --

---------------------------------------------------------------------
OK! All cases for Question 3 unlocked.

Cannot backup when running ok with --local.
```

思路：当骰子数量为`0`时，使用`Piggy Points`规则；否则，正常摇骰子。

实现代码如下：

```
def take_turn(num_rolls, opponent_score, dice=six_sided, goal=GOAL_SCORE):
    """Simulate a turn rolling NUM_ROLLS dice, which may be 0 in the case
    of a player using Piggy Points.
    Return the points scored for the turn by the current player.
```

```
    num_rolls:       The number of dice rolls that will be made.
    opponent_score:  The total score of the opponent.
    dice:            A function that simulates a single dice roll outcome.
    goal:            The goal score of the game.
    """
    # Leave these assert statements here; they help check for errors.
    assert type(num_rolls) == int, 'num_rolls must be an integer.'
    assert num_rolls >= 0, 'Cannot roll a negative number of dice in take_turn.'
    assert num_rolls <= 10, 'Cannot roll more than 10 dice.'
    assert opponent_score < goal, 'The game should be over.'
    # BEGIN PROBLEM 3
    "*** YOUR CODE HERE ***"
    if num_rolls == 0:
        return piggy_points(opponent_score)
    else:
        return roll_dice(num_rolls, dice)
    # END PROBLEM 3
```

可以使用如下命令进行测评：

```
python3 ok -q 03 --local
```

Problem 4 (2 pt)

实现`more_boar(player_score, opponent_score)`函数，`player_score`是当前玩家的分数，`opponent_score`是对手的分数，根据`More Boar`规则，判断下一个回合属于哪个玩家，若属于当前玩家，则返回`True`，否则返回`False`。

运行如下代码进行解锁测评：

```
python3 ok -q 04 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 3 > Suite 1 > Case 1
(cases remaining: 10)

>>> from hog import *
>>> take_turn(2, 0, make_test_dice(4, 5, 1))
? 9
-- OK! --
```

```
---------------------------------------------------------------------
Question 3 > Suite 1 > Case 2
(cases remaining: 9)

>>> from hog import *
>>> take_turn(3, 0, make_test_dice(4, 6, 1))
? 1
-- OK! --

---------------------------------------------------------------------
Question 3 > Suite 1 > Case 3
(cases remaining: 8)

>>> from hog import *
>>> take_turn(0, 2)
? 7
-- OK! --

---------------------------------------------------------------------
Question 3 > Suite 1 > Case 4
(cases remaining: 7)

>>> from hog import *
>>> take_turn(0, 0)
? 3
-- OK! --

---------------------------------------------------------------------
OK! All cases for Question 4 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下:

```python
def more_boar(player_score, opponent_score):
    """Return whether the player gets an extra turn.

    player_score:   The total score of the current player.
    opponent_score: The total score of the other player.

    >>> more_boar(21, 43)
    True
    >>> more_boar(22, 43)
    True
    >>> more_boar(43, 21)
    False
    >>> more_boar(12, 12)
    False
    >>> more_boar(7, 8)
    False
    """
    # BEGIN PROBLEM 4
    "*** YOUR CODE HERE ***"
    def two_digits(x):
        a = x % 10
        b = 0

        if x < 10:
            return b, a

        while x > 0:
            b = a
            a = x % 10
            x //= 10

        return a, b

    tmp_p1, tmp_p2 = two_digits(player_score)
    tmp_o1, tmp_o2 = two_digits(opponent_score)

    return tmp_p1 < tmp_o1 and tmp_p2 < tmp_o2
    # END PROBLEM 4
```

可以使用如下命令进行测评：

```
python3 ok -q 04 --local
```

Problem 5 (3 pt)

实现`play`函数，模拟游戏的运行，返回两个玩家的最终分数：

- `strategy0`：玩家0的策略，根据当前双方的分数，返回骰子数量
- `strategy1`：玩家1的策略，根据当前双方的分数，返回骰子数量
- `score0`：玩家0的分数
- `score1`：玩家1的分数
- `dice`：骰子函数
- `goal`：目标分数

运行如下代码进行解锁测评：

```
python3 ok -q 05 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 5 > Suite 1 > Case 1
(cases remaining: 111)
```

```
Q: The variables score0 and score1 are the scores for Player 0
and Player 1, respectively. Under what conditions should the
game continue?
Choose the number of the correct choice:
0) While score1 is less than goal
1) While at least one of score0 or score1 is less than goal
2) While score0 and score1 are both less than goal
3) While score0 is less than goal
? 2
-- OK! --

---------------------------------------------------------------------
Question 5 > Suite 1 > Case 2
(cases remaining: 110)

Q: What is a strategy in the context of this game?
Choose the number of the correct choice:
0) A player's desired turn outcome
1) A function that returns the number of dice a player will roll
2) The number of dice a player will roll
? 1
-- OK! --

---------------------------------------------------------------------
Question 5 > Suite 1 > Case 3
(cases remaining: 109)

Q: If strategy1 is Player 1's strategy function, score0 is
Player 0's current score, and score1 is Player 1's current
score, then which of the following demonstrates correct
usage of strategy1?
Choose the number of the correct choice:
0) strategy1(score1)
1) strategy1(score0, score1)
2) strategy1(score0)
3) strategy1(score1, score0)
? 1
-- Not quite. Try again! --

Choose the number of the correct choice:
0) strategy1(score1)
1) strategy1(score0, score1)
2) strategy1(score0)
3) strategy1(score1, score0)
? 3
-- OK! --

---------------------------------------------------------------------
Question 5 > Suite 2 > Case 1
(cases remaining: 108)

>>> import hog
>>> always_three = hog.make_test_dice(3)
>>> always = hog.always_roll
>>> #
>>> # Play function stops at goal
>>> s0, s1 = hog.play(always(5), always(3), score0=91, score1=10,
dice=always_three)
>>> s0
? 15
-- Not quite. Try again! --

? 106
-- OK! --

>>> s1
? 10
-- OK! --

---------------------------------------------------------------------
Question 5 > Suite 2 > Case 2
(cases remaining: 107)

>>> import hog
>>> always_three = hog.make_test_dice(3)
>>> always = hog.always_roll
>>> #
>>> # Goal score is not hardwired
>>> s0, s1 = hog.play(always(5), always(5), goal=10, dice=always_three)
>>> s0
? 15
-- OK! --

>>> s1
? 0
-- OK! --

---------------------------------------------------------------------
Question 5 > Suite 2 > Case 3
(cases remaining: 106)

-- Already unlocked --

---------------------------------------------------------------------
Question 5 > Suite 2 > Case 4
(cases remaining: 105)

-- Already unlocked --

---------------------------------------------------------------------
Question 5 > Suite 3 > Case 1
(cases remaining: 104)

>>> import hog
>>> always_three = hog.make_test_dice(3)
>>> always_seven = hog.make_test_dice(7)
>>> #
>>> # Use strategies
>>> # We recommend working this out turn-by-turn on a piece of paper (use Python
for difficult calculations).
>>> strat0 = lambda score, opponent: opponent % 10
>>> strat1 = lambda score, opponent: max((score // 10) - 4, 0)
>>> s0, s1 = hog.play(strat0, strat1, score0=71, score1=80, dice=always_seven)
>>> s0
? 74
-- OK! --

>>> s1
? 108
-- OK! --

---------------------------------------------------------------------
OK! All cases for Question 5 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下:

```
def play(strategy0, strategy1, score0=0, score1=0, dice=six_sided,
         goal=GOAL_SCORE, say=silence):
    """Simulate a game and return the final scores of both players, with Player
    0's score first, and Player 1's score second.

    A strategy is a function that takes two total scores as arguments (the
    current player's score, and the opponent's score), and returns a number of
    dice that the current player will roll this turn.

    strategy0:  The strategy function for Player 0, who plays first.
    strategy1:  The strategy function for Player 1, who plays second.
    score0:     Starting score for Player 0
    score1:     Starting score for Player 1
    dice:       A function of zero arguments that simulates a dice roll.
    goal:       The game ends and someone wins when this score is reached.
    say:        The commentary function to call at the end of the first turn.
    """
    who = 0  # Who is about to take a turn, 0 (first) or 1 (second)
    # BEGIN PROBLEM 5
    "*** YOUR CODE HERE ***"
    # END PROBLEM 5
    # (note that the indentation for the problem 6 prompt (***YOUR CODE HERE***)
might be misleading)
    # BEGIN PROBLEM 6
    "*** YOUR CODE HERE ***"
    while score0 < goal and score1 < goal:
        if who == 0:
            num_rolls = strategy0(score0, score1)
            score0 += take_turn(num_rolls, score1, dice, goal)
            if not more_boar(score0, score1):
                who = next_player(who)
        else:
            num_rolls = strategy1(score1, score0)
            score1 += take_turn(num_rolls, score0, dice, goal)
            if not more_boar(score1, score0):
                who = next_player(who)
    # END PROBLEM 6
    return score0, score1
```

可以使用如下命令进行测评：

```
python3 ok -q 05 --local
```

Phase 2: Commentary

Commentary examples

`say_scores`函数：打印两位玩家的分数，返回自身函数

```
def say_scores(score0, score1):
    """A commentary function that announces the score for each player."""
    print("Player 0 now has", score0, "and Player 1 now has", score1)
    return say_scores
```

`announce_lead_changes`：打印分数领先的玩家和分数差，返回一个评论函数`say`

```
def announce_lead_changes(last_leader=None):
    """Return a commentary function that announces lead changes.

    >>> f0 = announce_lead_changes()
    >>> f1 = f0(5, 0)
    Player 0 takes the lead by 5
    >>> f2 = f1(5, 12)
    Player 1 takes the lead by 7
    >>> f3 = f2(8, 12)
    >>> f4 = f3(8, 13)
    >>> f5 = f4(15, 13)
    Player 0 takes the lead by 2
    """
    def say(score0, score1):
        if score0 > score1:
            leader = 0
        elif score1 > score0:
            leader = 1
        else:
            leader = None
        if leader != None and leader != last_leader:
            print('Player', leader, 'takes the lead by', abs(score0 - score1))
        return announce_lead_changes(leader)
    return say
```

`both(f, g)`函数：返回一个能够打印`f`和`g`函数的评论的函数

```
def both(f, g):
    """Return a commentary function that says what f says, then what g says.

    NOTE: the following game is not possible under the rules, it's just
    an example for the sake of the doctest
    >>> h0 = both(say_scores, announce_lead_changes())
    >>> h1 = h0(10, 0)
    Player 0 now has 10 and Player 1 now has 0
    Player 0 takes the lead by 10
    >>> h2 = h1(10, 8)
    Player 0 now has 10 and Player 1 now has 8
    >>> h3 = h2(10, 17)
    Player 0 now has 10 and Player 1 now has 17
    Player 1 takes the lead by 7
    """
    def say(score0, score1):
        return both(f(score0, score1), g(score0, score1))
    return say
```

Problem 6 (2 pt)

在`play`函数中，每回合结束时调用`say`函数，下一回合掉用本回合`say`的返回值

运行如下代码进行解锁测评：

```
python3 ok -q 06 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 6 > Suite 1 > Case 1
(cases remaining: 8)

Q: What does a commentary function return?
Choose the number of the correct choice:
0) An integer representing the score.
1) None.
2) Another commentary function.
? 2
-- OK! --

---------------------------------------------------------------------
Question 6 > Suite 2 > Case 1
(cases remaining: 7)

>>> from hog import play, always_roll
>>> from dice import make_test_dice
>>> #
```

```
>>> def echo(s0, s1):
...     print(s0, s1)
...     return echo
>>> s0, s1 = play(always_roll(1), always_roll(1), dice=make_test_dice(3),
goal=5, say=echo)
(line 1)? 4 3
-- Not quite. Try again! --

(line 1)? 3 0
(line 2)? 3 3
(line 3)? 6 3
-- OK! --

---------------------------------------------------------------------
Question 6 > Suite 2 > Case 2
(cases remaining: 6)

>>> from hog import play, always_roll
>>> from dice import make_test_dice
>>> def count(n):
...     def say(s0, s1):
...         print(n, s0)
...         return count(n + 1)
...     return say
>>> s0, s1 = play(always_roll(1), always_roll(1), dice=make_test_dice(5),
goal=10, say=count(1))
(line 1)? 1 5
(line 2)? 2 5
(line 3)? 3 10
-- OK! --

---------------------------------------------------------------------
Question 6 > Suite 2 > Case 3
(cases remaining: 5)

>>> from hog import play, always_roll
>>> from dice import make_test_dice
>>> #
>>> def echo(s0, s1):
...     print(s0, s1)
...     return echo
>>> strat0 = lambda score, opponent: 1 - opponent // 10
>>> strat1 = always_roll(3)
>>> s0, s1 = play(strat0, strat1, dice=make_test_dice(4, 2, 6), goal=15,
say=echo)
(line 1)? 4 0
(line 2)? 4 12
(line 3)? 6 12
-- Not quite. Try again! --

(line 1)? 4 0
(line 2)? 4 12
(line 3)? 8 12
(line 4)? 8 24
-- OK! --
```

```
---------------------------------------------------------------------
Question 6 > Suite 2 > Case 4
(cases remaining: 4)

>>> from hog import play, always_roll
>>> from dice import make_test_dice
>>> #
>>> # Ensure that say is properly updated within the body of play
>>> def total(s0, s1):
...     print(s0 + s1)
...     return echo
>>> def echo(s0, s1):
...     print(s0, s1)
...     return total
>>> s0, s1 = play(always_roll(1), always_roll(1), dice=make_test_dice(2, 5),
goal=10, say=echo)
(line 1)? 2 0
(line 2)? 7
(line 3)? 4 5
(line 4)? 14
-- OK! --

---------------------------------------------------------------------
Question 6 > Suite 2 > Case 5
(cases remaining: 3)

>>> from hog import play, always_roll
>>> from dice import make_test_dice
>>> #
>>> # Ensure that say is properly updated within the body of play even with extra
turns
>>> def total(s0, s1):
...     print(s0 + s1)
...     return echo
>>> def echo(s0, s1):
...     print(s0, s1)
...     return total
>>> s0, s1 = play(always_roll(1), always_roll(1), dice=make_test_dice(8, 2),
goal=20, say=echo)
(line 1)? 8 0
(line 2)? 10
(line 3)? 16 2
(line 4)? 20
(line 5)? 16 12
(line 6)? 30
(line 7)? 18 20
-- OK! --

---------------------------------------------------------------------
Question 6 > Suite 3 > Case 1
(cases remaining: 2)

>>> from hog import play, always_roll, both, announce_lead_changes, say_scores
>>> from dice import make_test_dice
>>> #
>>> def echo_0(s0, s1):
...     print('*', s0)
...     return echo_0
>>> def echo_1(s0, s1):
...     print('**', s1)
...     return echo_1
>>> s0, s1 = play(always_roll(1), always_roll(1), dice=make_test_dice(2),
goal=3, say=both(echo_0, echo_1))
(line 1)? * 2
(line 2)? ** 0
(line 3)? * 2
(line 4)? ** 2
(line 5)? * 4
(line 6)? ** 2
-- OK! --

---------------------------------------------------------------------
Question 6 > Suite 3 > Case 2
(cases remaining: 1)

-- Already unlocked --

---------------------------------------------------------------------
OK! All cases for Question 6 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下，在`play`函数中加入一行`say = say(score0, score1)`：

```
def play(strategy0, strategy1, score0=0, score1=0, dice=six_sided,
         goal=GOAL_SCORE, say=silence):
    """Simulate a game and return the final scores of both players, with Player
    0's score first, and Player 1's score second.

    A strategy is a function that takes two total scores as arguments (the
    current player's score, and the opponent's score), and returns a number of
    dice that the current player will roll this turn.

    strategy0:  The strategy function for Player 0, who plays first.
    strategy1:  The strategy function for Player 1, who plays second.
    score0:     Starting score for Player 0
    score1:     Starting score for Player 1
    dice:       A function of zero arguments that simulates a dice roll.
    goal:       The game ends and someone wins when this score is reached.
    say:        The commentary function to call at the end of the first turn.
    """
    who = 0  # Who is about to take a turn, 0 (first) or 1 (second)
    # BEGIN PROBLEM 5
    "*** YOUR CODE HERE ***"
    # END PROBLEM 5
    # (note that the indentation for the problem 6 prompt (***YOUR CODE HERE***)
might be misleading)
    # BEGIN PROBLEM 6
    "*** YOUR CODE HERE ***"
    while score0 < goal and score1 < goal:
        if who == 0:
            num_rolls = strategy0(score0, score1)
            score0 += take_turn(num_rolls, score1, dice, goal)
            if not more_boar(score0, score1):
                who = next_player(who)
        else:
            num_rolls = strategy1(score1, score0)
            score1 += take_turn(num_rolls, score0, dice, goal)
            if not more_boar(score1, score0):
                who = next_player(who)
        say = say(score0, score1)   # here!
    # END PROBLEM 6
    return score0, score1
```

可以使用如下命令进行测评：

```
python3 ok -q 06 --local
```

Problem 7 (3 pt)

实现`announce_highest(who, last_score=0, running_high=0)`函数:

- `who`： `1`表示玩家1， `0`表示玩家0
- `last_score`： 上一个分数
- `running_high`： 分数的最高增长

运行如下代码进行解锁测评：

```
python3 ok -q 07 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 7 > Suite 1 > Case 1
(cases remaining: 5)

Q: What does announce_highest return?
Choose the number of the correct choice:
0) A string containing the largest point increase for the
   current player.
1) The current largest point increase between both
   players.
```

```
2) A commentary function that prints information about the
   biggest point increase for the current player.
? 2
-- OK! --

---------------------------------------------------------------------
Question 7 > Suite 1 > Case 2
(cases remaining: 4)

Q: When does the commentary function returned by announce_highest
print something out?
Choose the number of the correct choice:
0) After each turn.
1) When the current player, given by the parameter `who`,
   earns the biggest point increase yet between both
   players in the game.
2) When the current player, given by the parameter `who`,
   earns their biggest point increase yet in the game.
? 1
-- Not quite. Try again! --

Choose the number of the correct choice:
0) After each turn.
1) When the current player, given by the parameter `who`,
   earns the biggest point increase yet between both
   players in the game.
2) When the current player, given by the parameter `who`,
   earns their biggest point increase yet in the game.
? 2
-- OK! --

---------------------------------------------------------------------
Question 7 > Suite 1 > Case 3
(cases remaining: 3)

Q: What does the parameter last_score represent?
Choose the number of the correct choice:
0) The relevant player's score before this turn.
1) The opponent's score before this turn.
2) The last highest gain for the current player.
? 0
-- OK! --

---------------------------------------------------------------------
Question 7 > Suite 2 > Case 1
(cases remaining: 2)

-- Already unlocked --

---------------------------------------------------------------------
Question 7 > Suite 2 > Case 2
(cases remaining: 1)

-- Already unlocked --
```

```
---------------------------------------------------------------------
OK! All cases for Question 7 unlocked.

Cannot backup when running ok with --local.
```

思路：返回一个函数`func`，打印评论时更新`last_score`和`running_high`，返回`announce_highest`函数的调用，新的函数的环境帧会有新的值，`nonlocal`关键字可以引入父环境中的引用。

实现代码如下：

```
def announce_highest(who, last_score=0, running_high=0):
    """Return a commentary function that announces when WHO's score
    increases by more than ever before in the game.

    NOTE: the following game is not possible under the rules, it's just
    an example for the sake of the doctest

    >>> f0 = announce_highest(1) # Only announce Player 1 score gains
    >>> f1 = f0(12, 0)
    >>> f2 = f1(12, 9)
    Player 1 has reached a new maximum point gain. 9 point(s)!
    >>> f3 = f2(20, 9)
    >>> f4 = f3(20, 30)
    Player 1 has reached a new maximum point gain. 21 point(s)!
    >>> f5 = f4(20, 47) # Player 1 gets 17 points; not enough for a new high
    >>> f6 = f5(21, 47)
    >>> f7 = f6(21, 77)
    Player 1 has reached a new maximum point gain. 30 point(s)!
    """
    assert who == 0 or who == 1, 'The who argument should indicate a player.'
    # BEGIN PROBLEM 7
    "*** YOUR CODE HERE ***"
    def func(score0, score1):
        nonlocal last_score
        nonlocal running_high
        ls = last_score
        rh = running_high
        if who == 0:
            tmp = score0 - ls
            if tmp > rh:
                print("Player 0 has reached a new maximum point gain.",
tmp,"point(s)!")
                rh = tmp
            ls = score0
        else:
            tmp = score1 - ls
            if tmp > rh:
                print("Player 1 has reached a new maximum point gain.",
tmp,"point(s)!")
                rh = tmp
            ls = score1
        return announce_highest(who, ls, rh)
    return func
    # END PROBLEM 7
```

可以使用如下命令进行测评：

```
python3 ok -q 07 --local
```

Phase 3: Strategies

Problem 8 (2 pt)

实现`make_averaged(original_function, trials_count=1000)`函数，返回一个函数，有和`original_function`一样的参数，调用后，计算调用`trials_count`次`original_function`函数的平均值。

运行如下代码进行解锁测评：

```
python3 ok -q 08 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 8 > Suite 1 > Case 1
(cases remaining: 7)

Q: What makes make_averaged a higher order function?
Choose the number of the correct choice:
0) It uses the *args keyword
1) It calls a function that is not itself
2) It contains a nested function
3) It both takes in a function as an argument and returns a function
? 3
-- OK! --

---------------------------------------------------------------------
Question 8 > Suite 1 > Case 2
(cases remaining: 6)

Q: How many arguments does the function passed into make_averaged take?
Choose the number of the correct choice:
0) An arbitrary amount, which is why we need to use *args to call it
1) Two
2) None
? 0
-- OK! --
```

```
---------------------------------------------------------------------
Question 8 > Suite 2 > Case 1
(cases remaining: 5)

>>> from hog import *
>>> dice = make_test_dice(3, 1, 5, 6)
>>> averaged_dice = make_averaged(dice, 1000)
>>> # Average of calling dice 1000 times
>>> averaged_dice()
? 3.75
-- OK! --

---------------------------------------------------------------------
Question 8 > Suite 2 > Case 2
(cases remaining: 4)

>>> from hog import *
>>> dice = make_test_dice(3, 1, 5, 6)
>>> averaged_roll_dice = make_averaged(roll_dice, 1000)
>>> # Average of calling roll_dice 1000 times
>>> # Enter a float (e.g. 1.0) instead of an integer
>>> averaged_roll_dice(2, dice)
? 7.5
-- Not quite. Try again! --

? 6
-- Not quite. Try again! --

? 6.0
-- OK! --

---------------------------------------------------------------------
Question 8 > Suite 3 > Case 1
(cases remaining: 3)

-- Already unlocked --

---------------------------------------------------------------------
Question 8 > Suite 3 > Case 2
(cases remaining: 2)

-- Already unlocked --

---------------------------------------------------------------------
Question 8 > Suite 3 > Case 3
(cases remaining: 1)

-- Already unlocked --

---------------------------------------------------------------------
OK! All cases for Question 8 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下：

```
def make_averaged(original_function, trials_count=1000):
    """Return a function that returns the average value of ORIGINAL_FUNCTION
    when called.

    To implement this function, you will have to use *args syntax, a new Python
    feature introduced in this project.  See the project description.

    >>> dice = make_test_dice(4, 2, 5, 1)
    >>> averaged_dice = make_averaged(roll_dice, 1000)
    >>> averaged_dice(1, dice)
    3.0
    """
    # BEGIN PROBLEM 8
    "*** YOUR CODE HERE ***"
    def func(*argv):
        sum = 0
        for i in range(0, trials_count):
            sum += original_function(*argv)
        return sum / trials_count
    return func
    # END PROBLEM 8
```

可以使用如下命令进行测评：

```
python3 ok -q 08 --local
```

Problem 9 (2 pt)

实现 `max_scoring_num_rolls` 函数，返回 `1~10` 中得分期望最大的骰子数量。

运行如下代码进行解锁测评：

```
python3 ok -q 09 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 9 > Suite 1 > Case 1
(cases remaining: 8)

Q: If multiple num_rolls are tied for the highest scoring
average, which should you return?
Choose the number of the correct choice:
```

```
0) The highest num_rolls
1) A random num_rolls
2) The lowest num_rolls
? 2
-- OK! --

---------------------------------------------------------------------
Question 9 > Suite 2 > Case 1
(cases remaining: 7)

>>> from hog import *
>>> dice = make_test_dice(3)   # dice always returns 3
>>> max_scoring_num_rolls(dice, trials_count=1000)
? 1
-- Not quite. Try again! --

? 10
-- OK! --

---------------------------------------------------------------------
Question 9 > Suite 2 > Case 2
(cases remaining: 6)

-- Already unlocked --

---------------------------------------------------------------------
Question 9 > Suite 2 > Case 3
(cases remaining: 5)

-- Already unlocked --

---------------------------------------------------------------------
Question 9 > Suite 3 > Case 1
(cases remaining: 4)

>>> from hog import *
>>> dice = make_test_dice(2)     # dice always rolls 2
>>> max_scoring_num_rolls(dice, trials_count=1000)
? 10
-- OK! --

---------------------------------------------------------------------
Question 9 > Suite 3 > Case 2
(cases remaining: 3)

>>> from hog import *
>>> dice = make_test_dice(1, 2)  # dice alternates 1 and 2
>>> max_scoring_num_rolls(dice, trials_count=1000)
? 1
-- OK! --

---------------------------------------------------------------------
Question 9 > Suite 3 > Case 3
(cases remaining: 2)
```

```
-- Already unlocked --

---------------------------------------------------------------------
Question 9 > Suite 3 > Case 4
(cases remaining: 1)

-- Already unlocked --

---------------------------------------------------------------------
OK! All cases for Question 9 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下：

```
def max_scoring_num_rolls(dice=six_sided, trials_count=1000):
    """Return the number of dice (1 to 10) that gives the highest average turn
score
    by calling roll_dice with the provided DICE a total of TRIALS_COUNT times.
    Assume that the dice always return positive outcomes.

    >>> dice = make_test_dice(1, 6)
    >>> max_scoring_num_rolls(dice)
    1
    """
    # BEGIN PROBLEM 9
    "*** YOUR CODE HERE ***"
    num_rolls = 0
    max_scores = 0
    for i in range(1, 11):
        averaged_dice = make_averaged(roll_dice, trials_count)
        tmp = averaged_dice(i, dice)
        if max_scores < tmp:
            num_rolls = i
            max_scores = tmp
    return num_rolls
    # END PROBLEM 9
```

可以使用如下命令进行测评：

```
python3 ok -q 09 --local
```

可以使用如下命令测试策略的胜率（`win rate`）：

```
python3 hog.py -r
```

结果如下：

```
Max scoring num rolls for six-sided dice: 5
always_roll(6) win rate: 0.4955
always_roll(8) win rate: 0.532
piggypoints_strategy win rate: 0.484
more_boar_strategy win rate: 0.51
final_strategy win rate: 0.5165
```

Problem 10 (1 pt)

实现`piggypoints_strategy(score, opponent_score, cutoff=8, num_rolls=6)`函数，如果使用`Piggy Points`规则至少能得到`cutoff`分数，就摇`0`个骰子。

运行如下代码进行解锁测评：

```
python3 ok -q 10 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 10 > Suite 1 > Case 1
(cases remaining: 106)

>>> from hog import *
>>> piggypoints_strategy(0, 3, cutoff=8, num_rolls=5)
? 0
-- OK! --

---------------------------------------------------------------------
Question 10 > Suite 1 > Case 2
(cases remaining: 105)

>>> from hog import *
>>> piggypoints_strategy(9, 0, cutoff=6, num_rolls=5)
? 5
-- OK! --

---------------------------------------------------------------------
Question 10 > Suite 1 > Case 3
(cases remaining: 104)

>>> from hog import *
>>> piggypoints_strategy(50, 2, cutoff=7, num_rolls=5)
? 0
```

```
-- OK! --

---------------------------------------------------------------------
Question 10 > Suite 1 > Case 4
(cases remaining: 103)

>>> from hog import *
>>> piggypoints_strategy(32, 0, cutoff=8, num_rolls=4)
? 4
-- OK! --

---------------------------------------------------------------------
Question 10 > Suite 1 > Case 5
(cases remaining: 102)

>>> from hog import *
>>> piggypoints_strategy(20, 1, cutoff=1, num_rolls=4)
? 0
-- OK! --

OK! All cases for Question 10 unlocked.

Cannot backup when running ok with --local.
```

实现代码如下：

```
def piggypoints_strategy(score, opponent_score, cutoff=8, num_rolls=6):
    """This strategy rolls 0 dice if that gives at least CUTOFF points, and
    rolls NUM_ROLLS otherwise.
    """
    # BEGIN PROBLEM 10
    return (0 if piggy_points(opponent_score) >= cutoff else num_rolls)
    # END PROBLEM 10
```

可以使用如下命令进行测评：

```
python3 ok -q 10 --local
```

Problem 11 (2 pt)

实现 `more_boar_strategy(score, opponent_score, cutoff=8, num_rolls=6)` 函数，如果能使用 `More Boar` 规则，就摇 `0` 个骰子；否则，若摇 `0` 个骰子得分大于 `cutoff`，就摇 `0` 个骰子，不然返回 `num_rolls`。

运行如下代码进行解锁测评：

```
python3 ok -q 11 -u --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
```

```
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Unlocking tests

At each "? ", type what you would expect the output to be.
Type exit() to quit

---------------------------------------------------------------------
Question 11 > Suite 1 > Case 1
(cases remaining: 105)

>>> from hog import *
>>> more_boar_strategy(3, 19, cutoff=8, num_rolls=6)
? 0
-- OK! --

---------------------------------------------------------------------
Question 11 > Suite 1 > Case 2
(cases remaining: 104)

>>> from hog import *
>>> more_boar_strategy(30, 54, cutoff=7, num_rolls=6)
? 0
-- Not quite. Try again! --

? 6
-- OK! --

---------------------------------------------------------------------
Question 11 > Suite 1 > Case 3
(cases remaining: 103)

>>> from hog import *
>>> more_boar_strategy(17, 36, cutoff=100, num_rolls=6)
? 6
-- Not quite. Try again! --

? 0
-- OK! --

---------------------------------------------------------------------
Question 11 > Suite 1 > Case 4
(cases remaining: 102)

>>> from hog import *
>>> more_boar_strategy(24, 3, cutoff=8, num_rolls=6)
? 0
-- OK! --

---------------------------------------------------------------------
OK! All cases for Question 11 unlocked.

Cannot backup when running ok with --local.
```

思路：注意到，检测`More Boar`规则要在加上`Piggy Points`规则的得分之后进行。

实现代码如下：

```
def more_boar_strategy(score, opponent_score, cutoff=8, num_rolls=6):
    """This strategy rolls 0 dice when it triggers an extra turn. It also
    rolls 0 dice if it gives at least CUTOFF points and does not give an extra
turn.
    Otherwise, it rolls NUM_ROLLS.
    """
    # BEGIN PROBLEM 11
    return 0 if more_boar(score + piggy_points(opponent_score), opponent_score)
else piggypoints_strategy(score, opponent_score, cutoff, num_rolls)
    # END PROBLEM 11
```

可以使用如下命令进行测评：

```
python3 ok -q 11 --local
```

Optional: Problem 12 (0 pt)

自己实现一个策略，追求更高胜率，有时间再来填坑。

实现代码如下：

```

```

可以使用如下命令进行测评：

```
python3 ok -q 12 --local
```

最终测评

可以使用如下命令进行测评：

```
python3 ok --local
```

结果如下：

```
=====================================================================
Assignment: Project 1: Hog
OK, version v1.18.1
=====================================================================

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    565 test cases passed! No cases failed.

Cannot backup when running ok with --local.
```