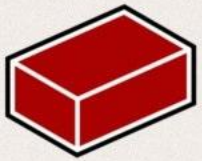
HackerClub



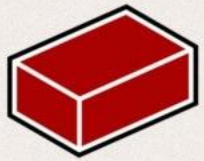
Redbrick

DCU's Networking Society

Today...

is not a deep dive into OOP.

is a brief guide on OOP for problem solving.



Redbrick

DCU's Networking Society

Log into Kattis

When logged in search for “Take Two Stones”.

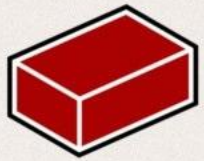
Open a new tab, and search for “pizza2” on Kattis.

Read it...

what is it asking you to find?

what data is it giving you to work with?

what objects can you identify in the description?



Redbrick

DCU's Networking Society

Find the nouns (objects and data attributes)

Alice and **Bob** are playing a new **game** of **stones**. There are *N* **stones** placed on the **ground**, forming a **sequence**. The stones are labeled from 1 to *N*.

Alice and **Bob** in turns take exactly two consecutive **stones** on the **ground** until there are no consecutive **stones** on the **ground**.

That is, each **player** can take **stone** *i* and **stone** $i + 1$, where $1 \leqslant i \leqslant N - 1$.

If the **number** of **stones** left is odd, **Alice** wins. Otherwise, **Bob** wins.

Assume both **Alice** and **Bob** play optimally and **Alice** plays first, do you know who the **winner** is?

Eliminate duplicates

Alice and **Bob** are playing a new **game** of **stones**. There are *N* stones placed on the **ground**, forming a **sequence**. The stones are labeled from 1 to *N*.

Alice and Bob in turns take exactly two consecutive stones on the ground until there are no consecutive stones on the ground.

That is, each **player** can take **stone** *i* and stone *i* + 1, where $1 \leqslant i \leqslant N - 1$.

If the **number** of **stones** left is odd, Alice wins. Otherwise, Bob wins.

Assume both Alice and Bob play optimally and Alice plays first, do you know who the **winner** is?

Classify

Alice and **Bob** are outputs, so we can call them attributes of the object.

The **number of stones** is the input so we can call it an attribute also.

The object these are all attributes of is the **game**. The game has **players** (**Alice** and **Bob**) and a **sequence of stones.**

The word **ground** can disregarded in this question.

Finally, our object needs a **winner**, which will be either **Alice** or **Bob**.

```python
class Game(object):
    def __init__(self, player1, player2, stones):
        self.player1 = player1
        self.player2 = player2
        self.stones = stones

    def get_winner(self):
        winner = self.player1 if self.stones % 2 != 0 else self.player2
        return winner

    def play(self):
        leftover = self.stones
        for i in range(1, self.stones):
            leftover -= 2
        self.stones = leftover


def main():
    n = int(input())
    game = Game('Alice', 'Bob', n)
    game.play()
    winner = game.get_winner()
    print(winner)


if __name__ == "__main__":
    main()
```

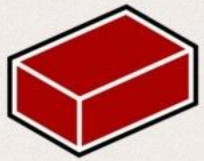
(1.)
(2.)
(3.)

(1.) The game uses 3 pieces of data, player1 **(Alice)**, player2 **(Bob)**, and the number of stones **(n)**.

We initialize the object using the data.

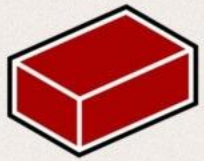
(2.) Calling .play() on the new game object will trigger an algorithm which follows hte rules specified in the problem description.

(3.) Finally we judge the winner on how many stones are leftover.



Redbrick
DCU's Networking Society

Questions?

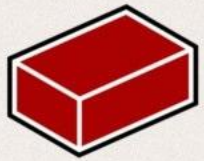


Redbrick
DCU's Networking Society

With that done…

Move onto your other tab, and try the “pizza2” problem.

I have a solution so… it can be done.



Redbrick

DCU’s Networking Society