Introduction to Data Science DS 2006 (15 credits)

Prof. Carlos N. Silla Jr.

Halmstad University

BATTLE OF DICES
ON TRIAL

Good News!

Since our Battle of Dices game went on trial, the player base has skyrocketed!

Currently one of the main requests by our player community is that we allow the game to be Multiplayer instead of only a 2 player game!

They have also asked that we allow them to identify themselves with their names/nicknames, so that they are able to take prints and brag about their amazing sheer luck power in social media!

Think and Reflect Time (by yourself)

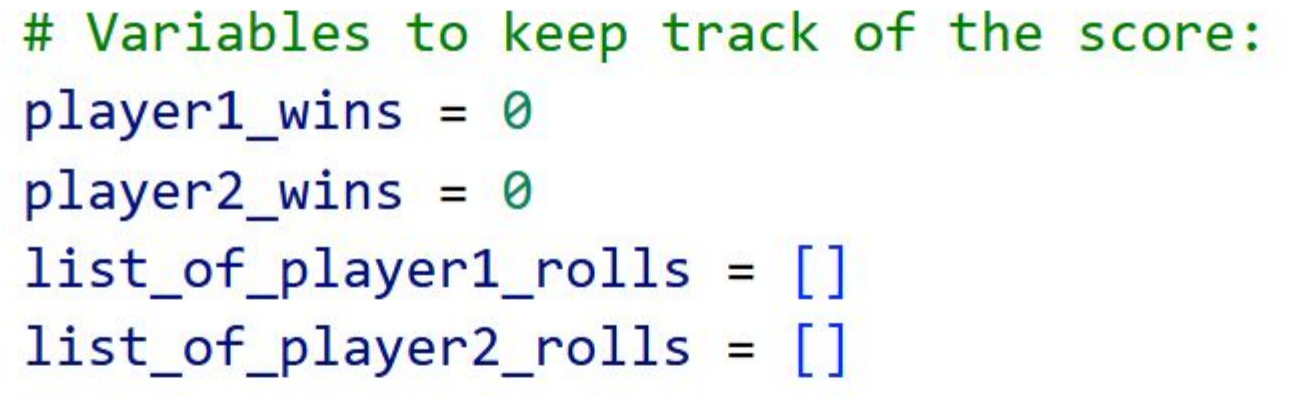
What are the main limitation(s) in your current code to implement the multiplayer version of the Battle of the dices?

Think and Reflect Time (in your teams)

What are the main limitation(s) in your current code to implement the multiplayer version of the Battle of the dices?

Main Limitation

Hard Coded Variables to Keep Track of the Scores / rolls for each individual player.



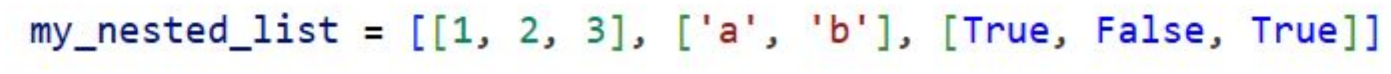
```
# Variables to keep track of the score:
player1_wins = 0
player2_wins = 0
list_of_player1_rolls = []
list_of_player2_rolls = []
```

Refactoring our Battle of Dices using Nested Lists

A Nested List is a List which contains another list in each position of its index.

Refactoring our Battle of Dices using Nested Lists

A Nested List is a List which contains another list in each position of its index.

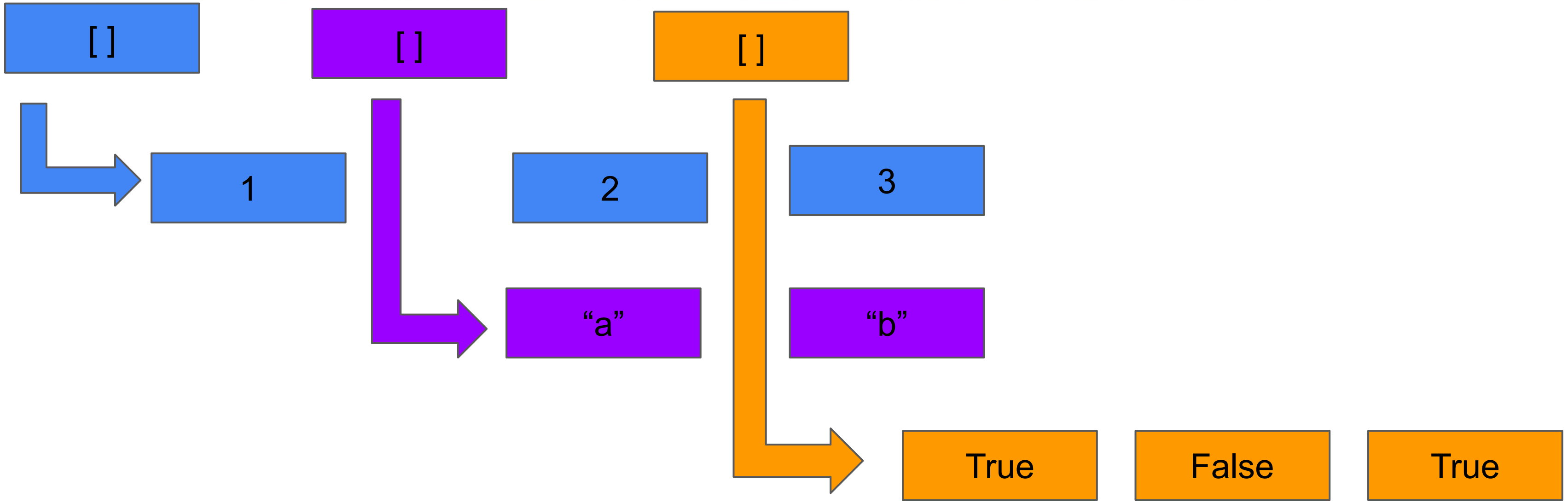


```
my_nested_list = [[1, 2, 3], ['a', 'b'], [True, False, True]]
```

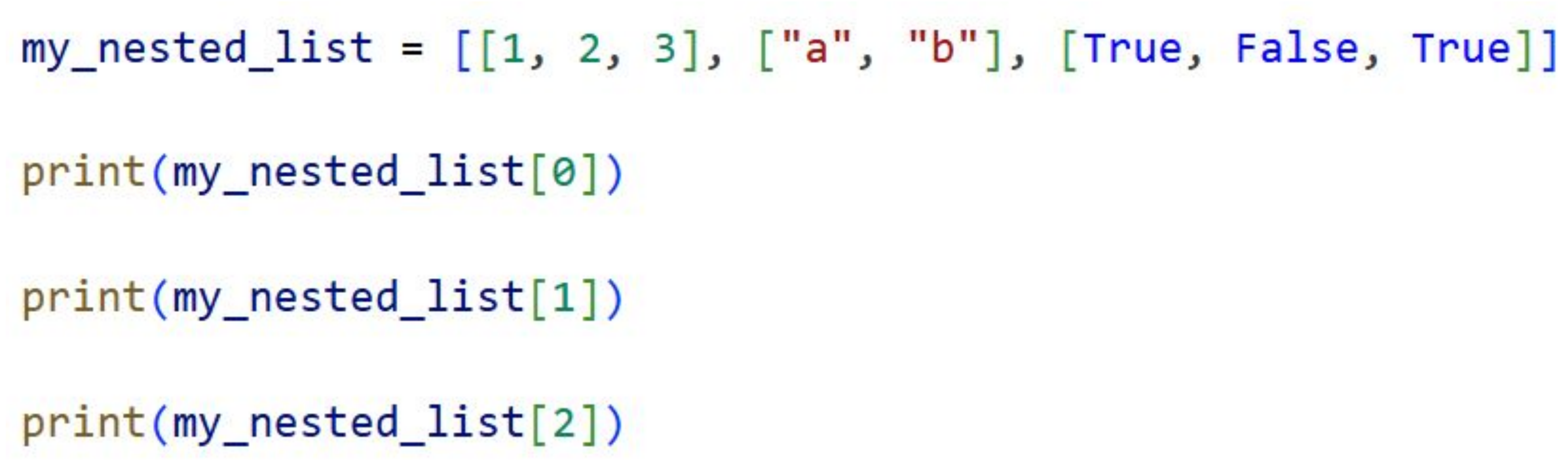
Refactoring our Battle of Dices using Nested Lists

A Nested List is a List which contains another list in each position of its index.

```
my_nested_list = [[1, 2, 3], ['a', 'b'], [True, False, True]]
```



Accessing Elements in a Nested Lists



```
my_nested_list = [[1, 2, 3], ["a", "b"], [True, False, True]]

print(my_nested_list[0])

print(my_nested_list[1])

print(my_nested_list[2])
```

Accessing Elements in a Nested Lists

```python
my_nested_list = [[1, 2, 3], ["a", "b"], [True, False, True]]

print(my_nested_list[0])

print(my_nested_list[1])

print(my_nested_list[2])
```

```
[1, 2, 3]

['a', 'b']

[True, False, True]
```

Accessing Specific Elements in the Nested Lists

```
my_nested_list = [[1, 2, 3], ["a", "b"], [True, False, True]]
```

```
print(my_nested_list[0][2])
```

```
3
```

```
print(my_nested_list[1][0])
```

```
a
```

```
print(my_nested_list[2][1])
```

```
False
```

Adding Elements to Nested Lists

```
my_nested_list = [[1, 2, 3], ["a", "b"], [True, False, True]]

my_nested_list[1].append("c")

print(my_nested_list[1])
```

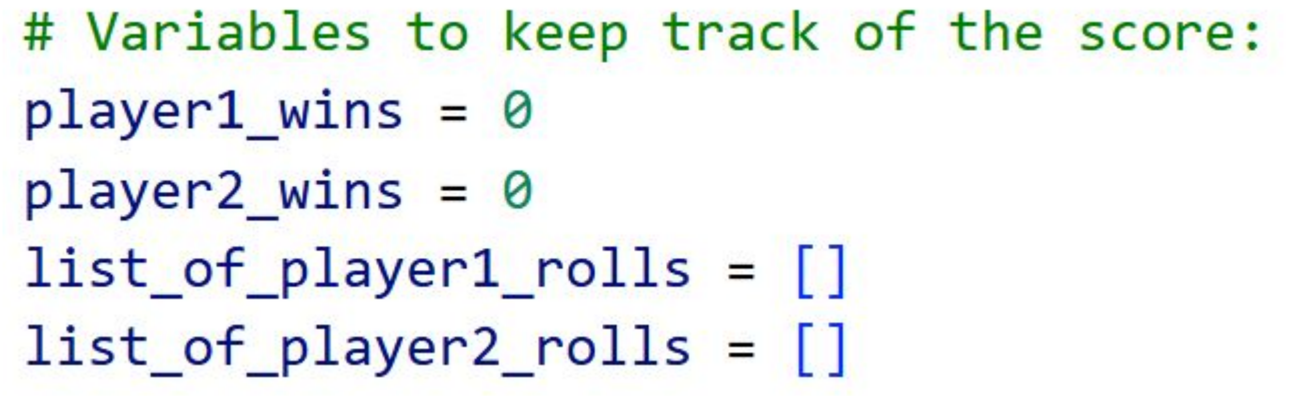
Adding Elements to Nested Lists

```python
my_nested_list = [[1, 2, 3], ["a", "b"], [True, False, True]]

my_nested_list[1].append("c")

print(my_nested_list[1])
```

```
['a', 'b', 'c']
```

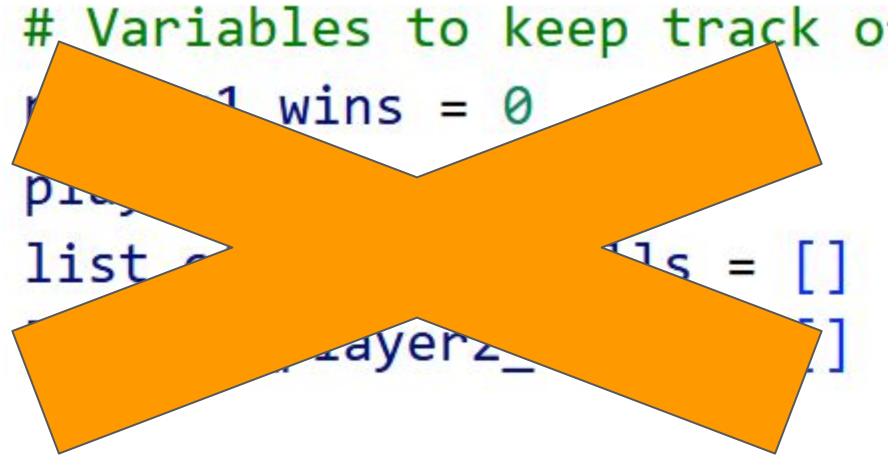
Kahoot Time!

Refactoring our Battle of Dices to use Nested Lists



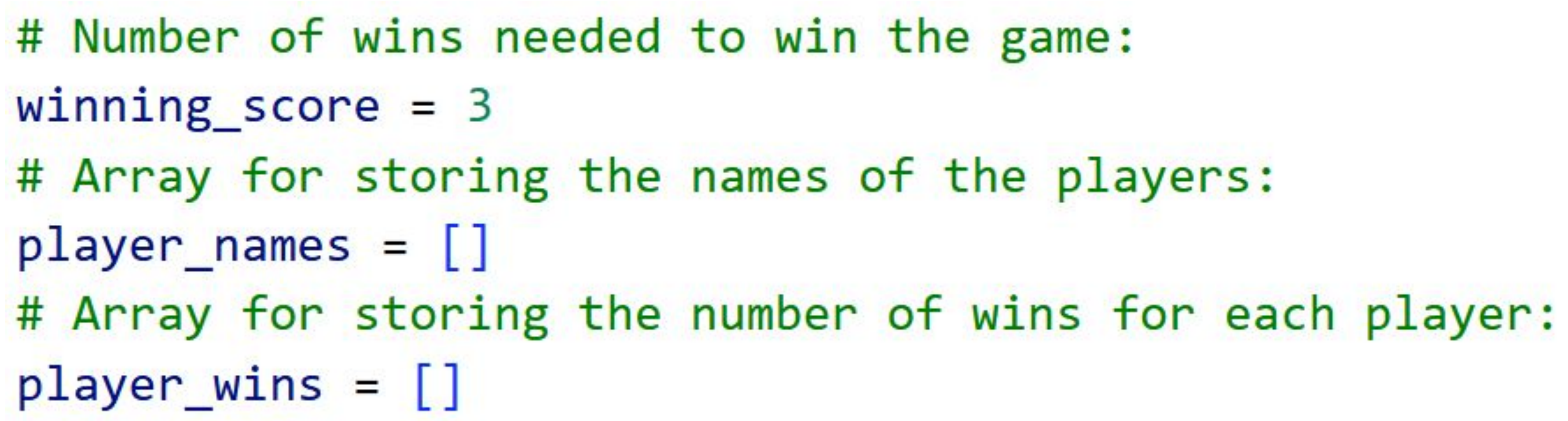
```
# Variables to keep track of the score:
player1_wins = 0
player2_wins = 0
list_of_player1_rolls = []
list_of_player2_rolls = []
```

Refactoring our Battle of Dices to use Nested Lists



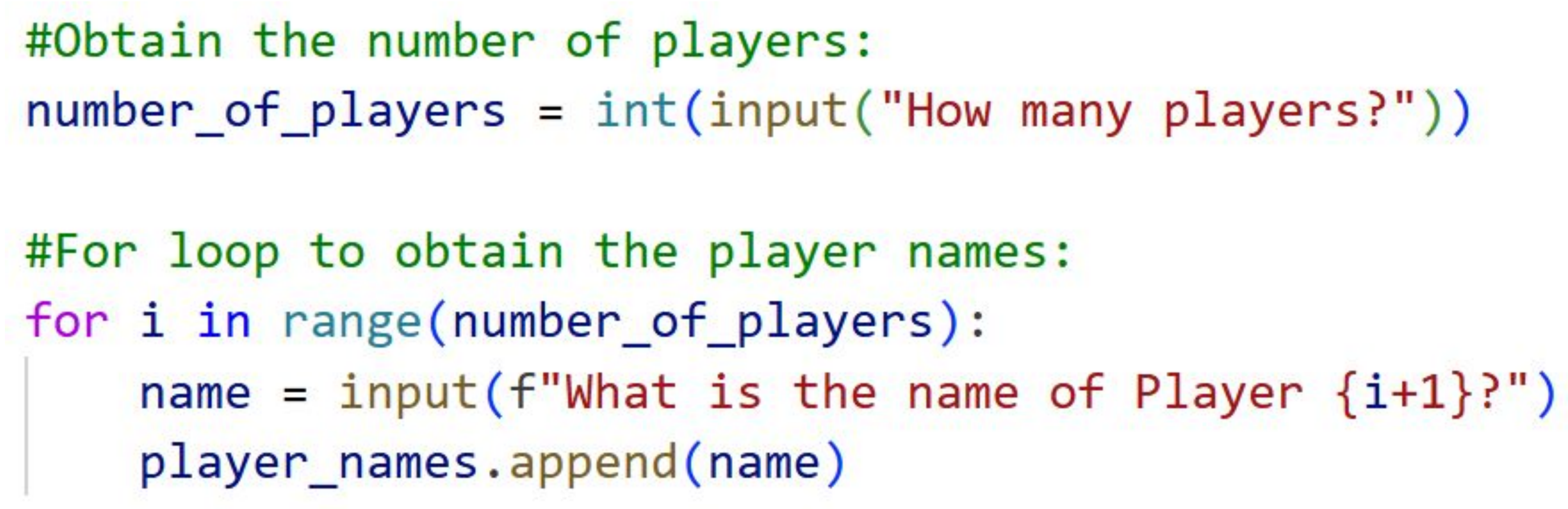
```
# Variables to keep track of the score:
[illegible]1_wins = 0
[illegible]
list_[illegible]lls = []
[illegible]layer2_[illegible][]
```

Refactoring our Battle of Dices to use Nested Lists



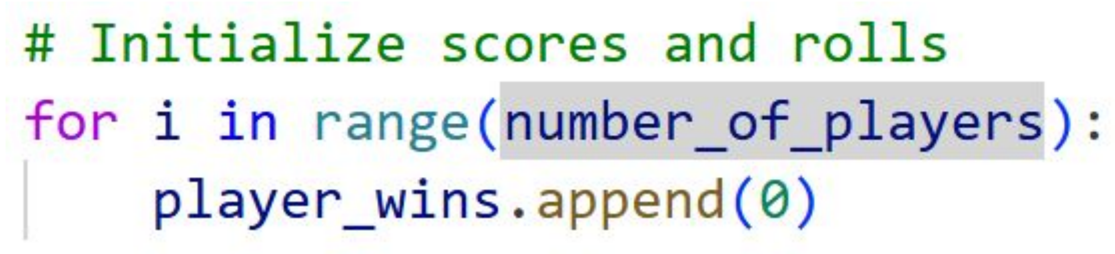
```
# Number of wins needed to win the game:
winning_score = 3
# Array for storing the names of the players:
player_names = []
# Array for storing the number of wins for each player:
player_wins = []
```

Getting the number of players and their names



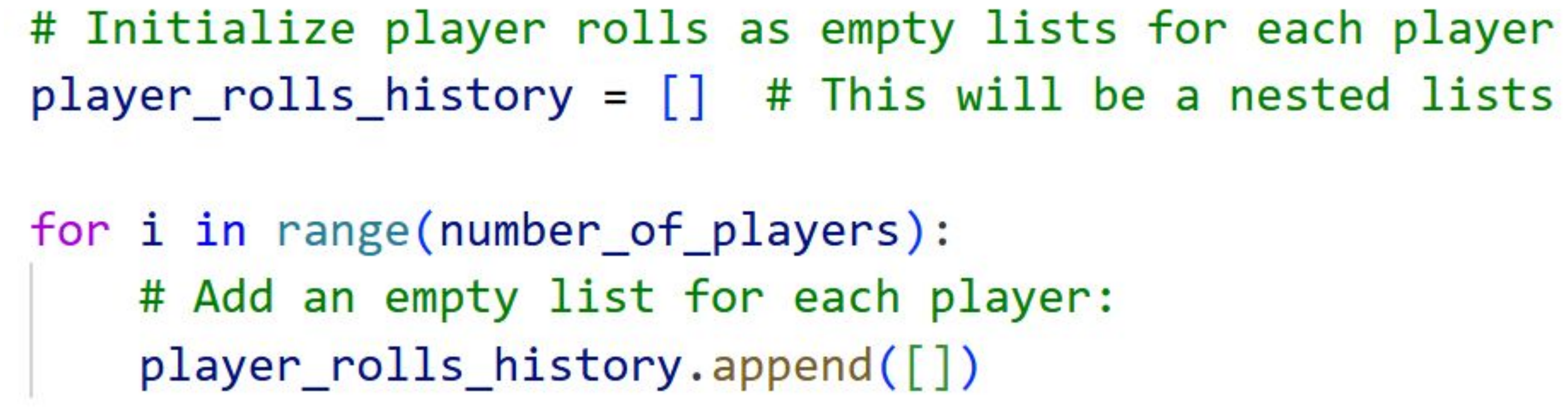
```python
#Obtain the number of players:
number_of_players = int(input("How many players?"))

#For loop to obtain the player names:
for i in range(number_of_players):
    name = input(f"What is the name of Player {i+1}?")
    player_names.append(name)
```

Initializing the List of Player Scores to have the value 0 in each position:

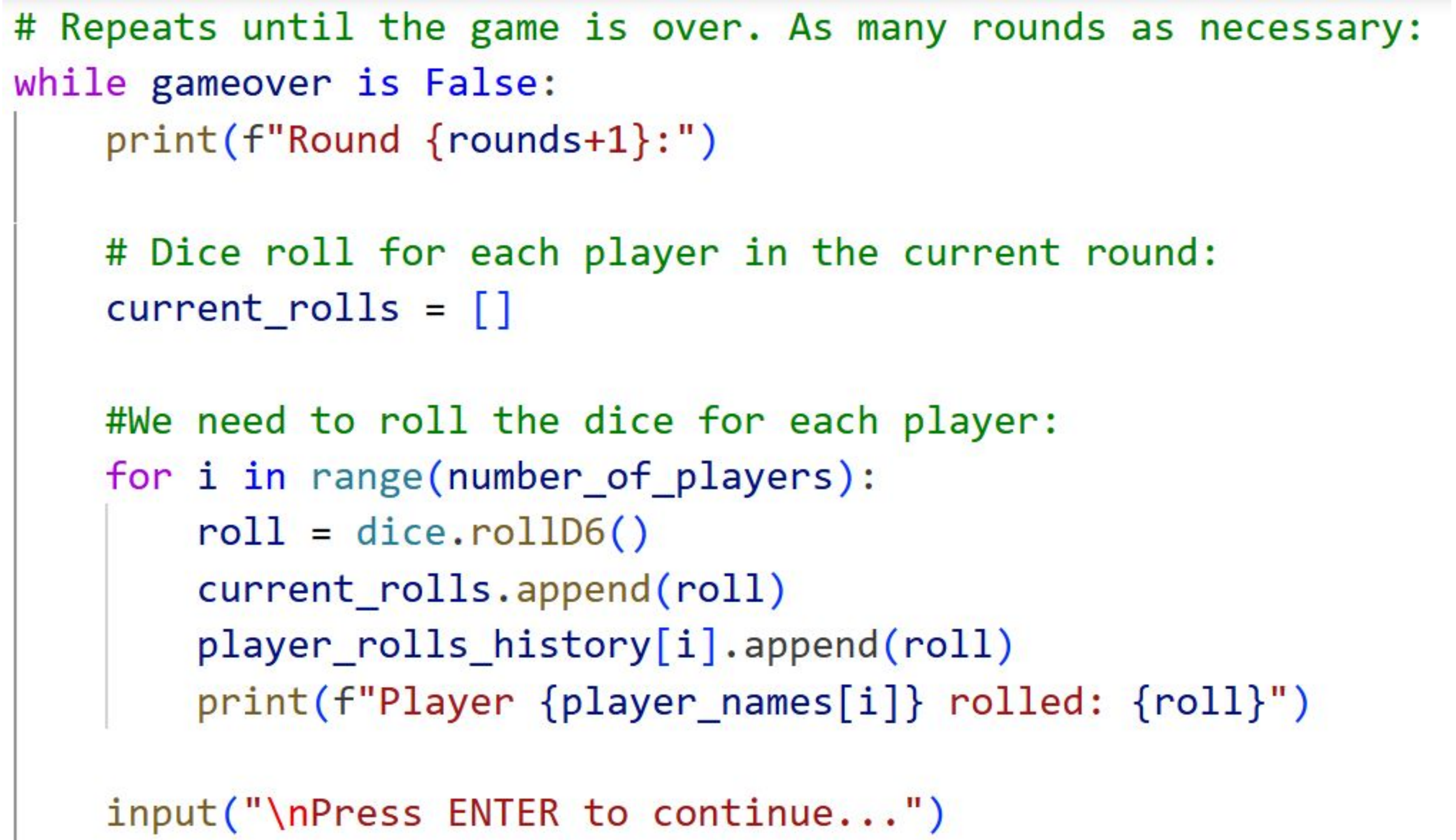


```python
# Initialize scores and rolls
for i in range(number_of_players):
    player_wins.append(0)
```

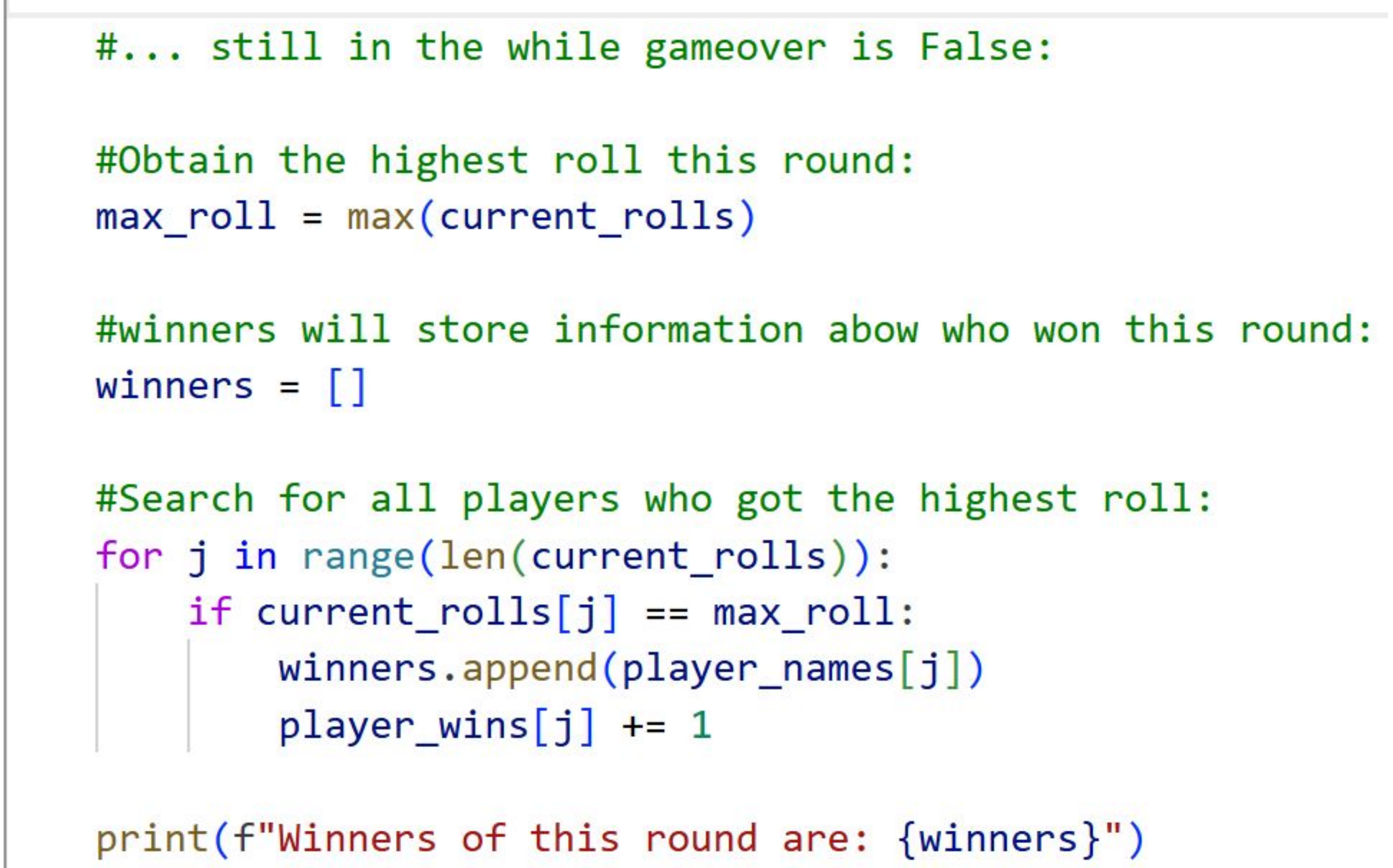
Initializing player rolls to have an empty list for each player:



```python
# Initialize player rolls as empty lists for each player
player_rolls_history = []  # This will be a nested lists

for i in range(number_of_players):
    # Add an empty list for each player:
    player_rolls_history.append([])
```



```python
# Repeats until the game is over. As many rounds as necessary:
while gameover is False:
    print(f"Round {rounds+1}:")

    # Dice roll for each player in the current round:
    current_rolls = []

    #We need to roll the dice for each player:
    for i in range(number_of_players):
        roll = dice.rollD6()
        current_rolls.append(roll)
        player_rolls_history[i].append(roll)
        print(f"Player {player_names[i]} rolled: {roll}")

    input("\nPress ENTER to continue...")
```



```
#... still in the while gameover is False:

#Obtain the highest roll this round:
max_roll = max(current_rolls)

#winners will store information abow who won this round:
winners = []

#Search for all players who got the highest roll:
for j in range(len(current_rolls)):
    if current_rolls[j] == max_roll:
        winners.append(player_names[j])
        player_wins[j] += 1

print(f"Winners of this round are: {winners}")
```

Definition and Usage

The `max()` function returns the item with the highest value, or the item with the highest value in an iterable.

If the values are strings, an alphabetically comparison is done.

Syntax

```
max(n1, n2, n3, ...)
```

Or:

```
max(iterable)
```

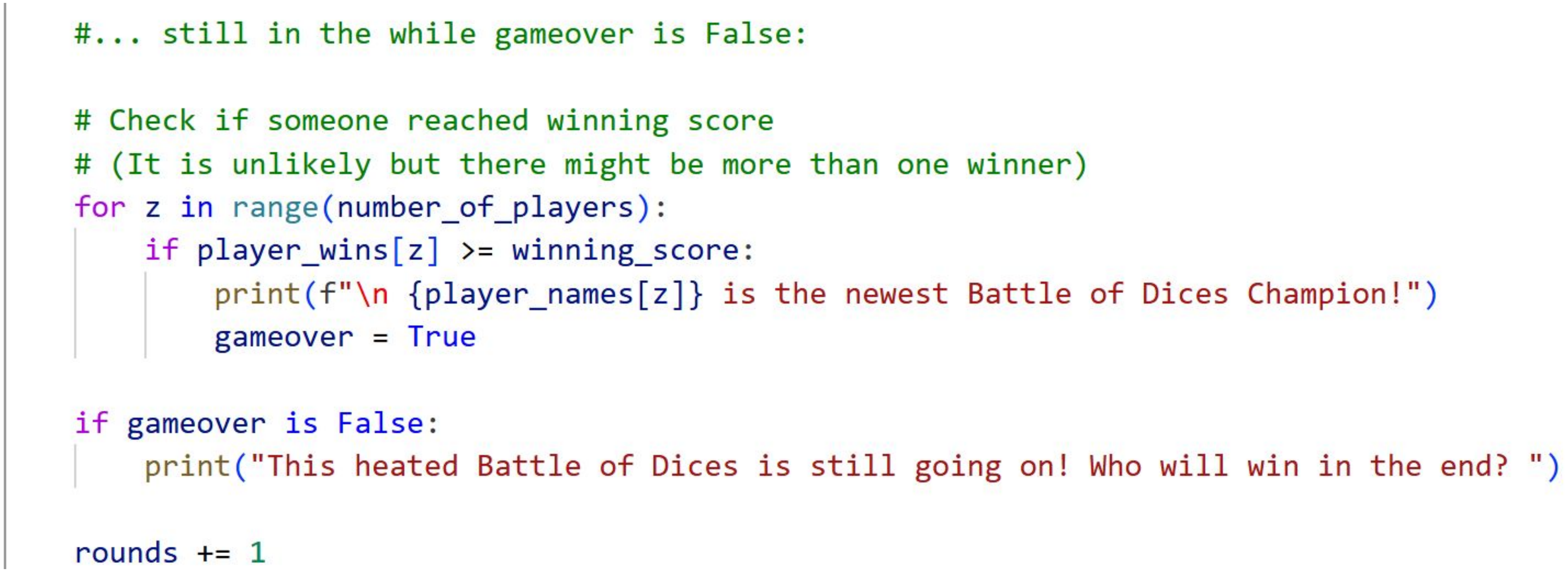
Parameter Values

Parameter	Description
n1, n2, n3, ...	One or more items to compare

Or:

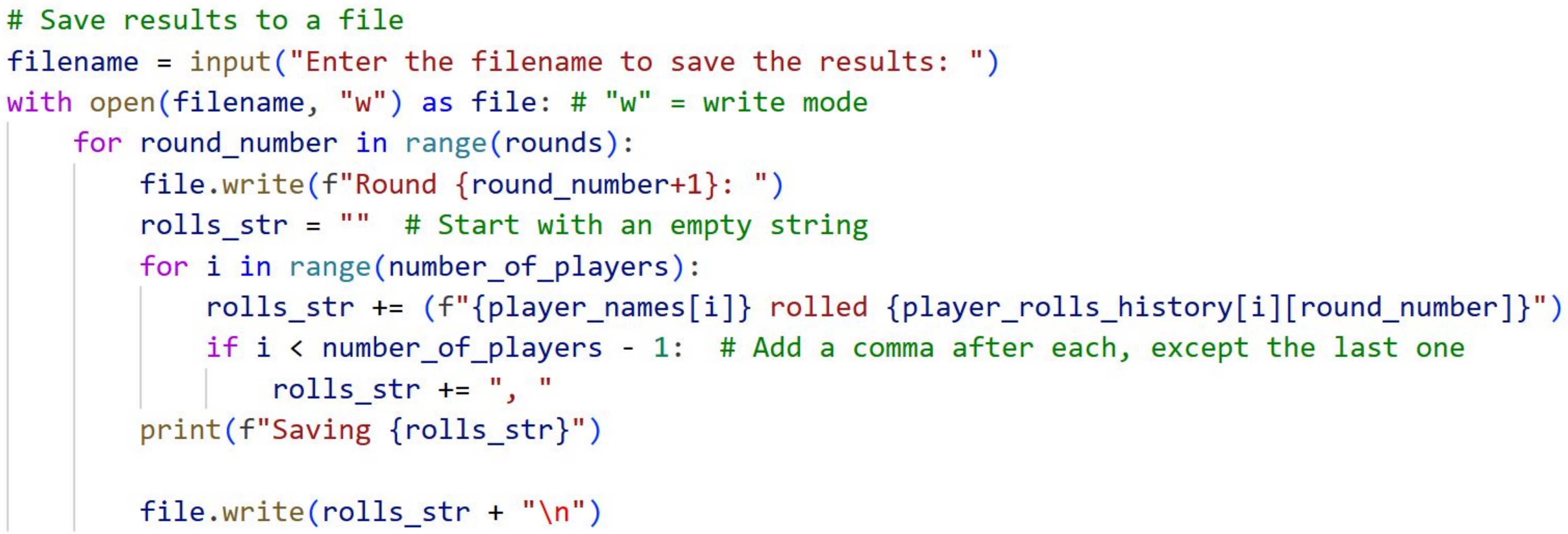
Parameter	Description
iterable	An iterable, with one or more items to compare

Check whether or not there were winning player(s):



```python
#... still in the while gameover is False:

# Check if someone reached winning score
# (It is unlikely but there might be more than one winner)
for z in range(number_of_players):
    if player_wins[z] >= winning_score:
        print(f"\n {player_names[z]} is the newest Battle of Dices Champion!")
        gameover = True

if gameover is False:
    print("This heated Battle of Dices is still going on! Who will win in the end? ")

rounds += 1
```

Save the results:



```python
# Save results to a file
filename = input("Enter the filename to save the results: ")
with open(filename, "w") as file: # "w" = write mode
    for round_number in range(rounds):
        file.write(f"Round {round_number+1}: ")
        rolls_str = ""  # Start with an empty string
        for i in range(number_of_players):
            rolls_str += (f"{player_names[i]} rolled {player_rolls_history[i][round_number]}")
            if i < number_of_players - 1:  # Add a comma after each, except the last one
                rolls_str += ", "
        print(f"Saving {rolls_str}")

        file.write(rolls_str + "\n")
```