CALLING FUNCTIONS AND METHODS

Python Programming — Auburn University

FREE FUNCTIONS

- Unlike Java, Python includes the notion of a sub-program that is not bound to an object or class.
- These are called “free functions” or just “functions.”
- We will learn to define our own functions soon but for now let’s look at calling some of the built-in functions.
- https://docs.python.org/3.9/library/functions.html

CALLING A FUNCTION: POSITIONAL ARGUMENTS

- Suppose there's a function called `some_function` that takes three arguments `x`, `y`, and `z`.
- Calling this function with "positional arguments":

  ```
  some_function(1,2,3)
  ```

- In this function call, `x` would be 1, y would be 2, and `z` would be 3. Hence the term "positional arguments."

CALLING A FUNCTION: POSITIONAL ARGUMENTS

Function's name

- Suppose there's a functio e_function that takes three arguments x, y, and z.
- Calling this function with "positional arguments":

```
    some_function(1,2,3)
```

- In this function call, x would be 1, y would be 2, and z would be 3. Hence the term "positional arguments."

CALLING A FUNCTION: POSITIONAL ARGUMENTS

Function's name

- Suppose there's a functio e_function that takes three arguments x, y, and z.
- Calling this function with "positional arguments":

```
some_function(1,2,3)
```

Arguments

- In this function call, x would be 2, and z would be 3. Hence the term "positional arguments

CALLING A FUNCTION: POSITIONAL ARGUMENTS

- Suppose there's a function called `some_function` that takes three arguments `x`, `y`, and `z`.
- Calling this function with "positional arguments":

  ```
  some_function(1,2,3)
  ```

- In this function call, `x` would be 1, y would be 2, and `z` would be 3. Hence the term "positional arguments."

CALLING A FUNCTION: KEYWORD ARGUMENTS

CALLING A FUNCTION: KEYWORD ARGUMENTS

- Calling this function with keyword arguments:

```
    some_function(x=1, y=2, z=3)
```

CALLING A FUNCTION: KEYWORD ARGUMENTS

- Calling this function with keyword arguments:
```
    some_function(x=1, y=2, z=3)
```

- Order doesn’t matter:
```
    some_function(z=3, x=1, y=2)
```

CALLING A FUNCTION: KEYWORD ARGUMENTS

- Calling this function with keyword arguments:
```
some_function(x=1, y=2, z=3)
```
- Order doesn't matter:
```
some_function(z=3, x=1, y=2)
```
- Keyword and positional argument forms can be mixed (positional first):
```
some_function(1, 2, z=3)
```

DEFAULT VALUES

- Some functions specify default values for argument.
- If a default value is supplied, that argument may be omitted when calling the function.
- Example: the “base” argument for the int() function defaults to 10.
- If the argument’s value is not specified, the default is used:

```
    int(’37’) -> 37                # base variable defaults to 10
    int(‘10111’, 2) -> 23          # base variable set to 2
```

DEFAULT VALUES: ANOTHER EXAMPLE

- The `input` function reads a line of text from the console.
- `inputs` can accept `0` or `1` arguments.
- Reading docs for `input:` https://docs.python.org/3.9/library/functions.html

VARIABLE ARITY FUNCTIONS

- The "arity" of a function is the number of arguments that it takes.
- Functions can be "variable arity" by specifying an argument whose name begins with "*". That argument will be given all unconsumed positional and non-keyword arguments.
- Reading docs for `print:` https://docs.python.org/3.9/library/functions.html
- `print` can take as many arguments as you like:

```
    print("Hello", name)
    print("abc", 1, 2, 3, "xyz")
    print()
```

- If you want to pass in values for the other arguments, you must pass them with keywords:

```
    print("Hello","Fred",sep='…')      prints Hello…Fred

    print("Hello",end='')
    print("world")                     prints Helloworld
```

OBJECTS AND METHODS

- Python is an OOPL, like Java. I'll assume you are generally familiar with objects, methods and sending messages.
- Many built-in Python functions return an object.
  - `open( )`, for example, returns a file object.
- Send messages to objects using the "." notation, just like Java:
  ```
  f = open("somefile.txt")
  first_line = f.readline()
  ```

OBJECTS AND METHODS

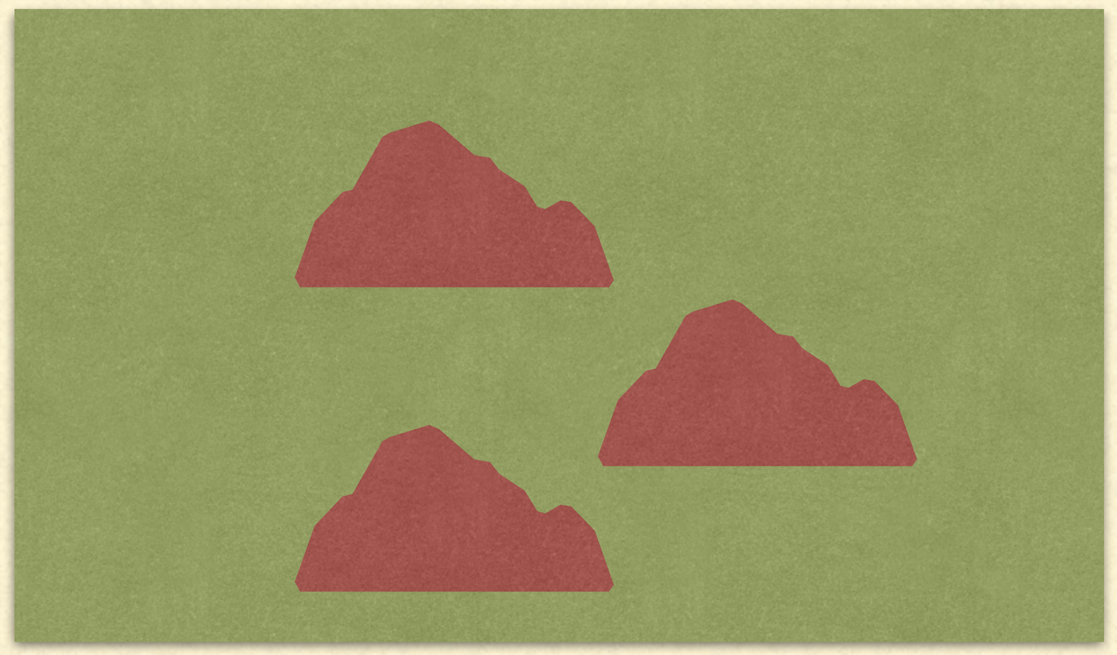
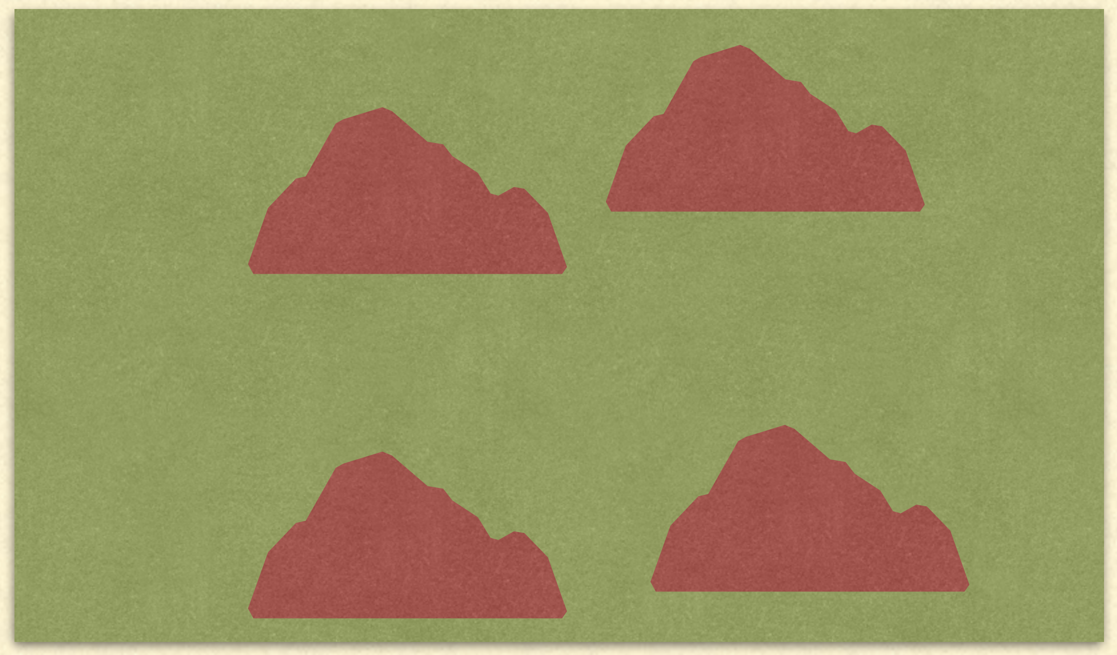
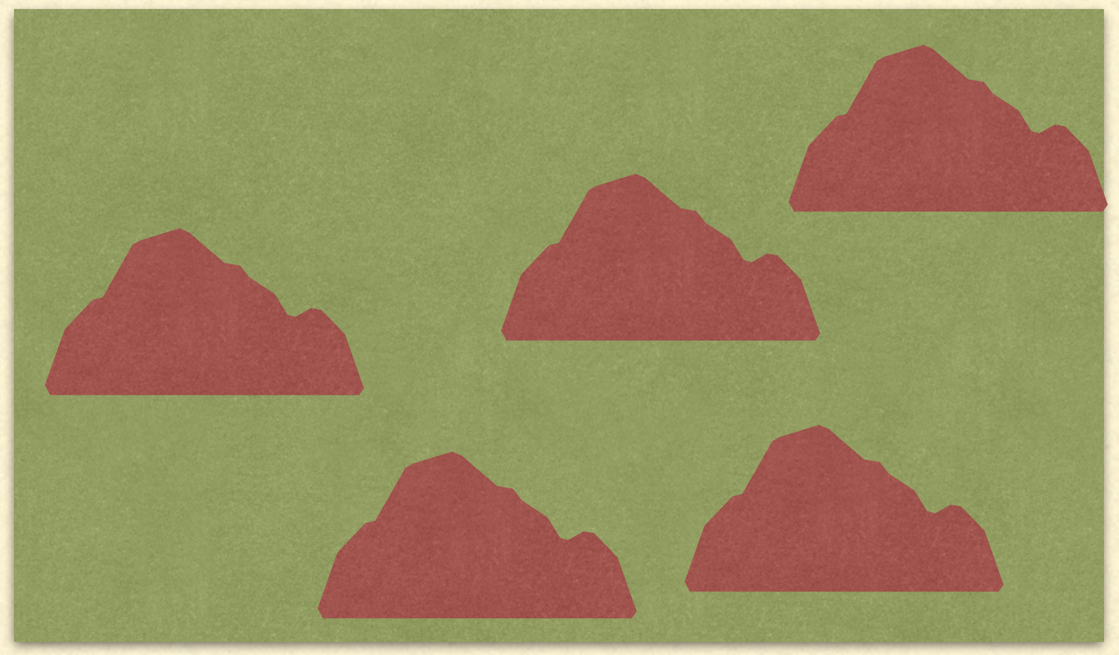
- Best way to see methods available for an object is through reference documentation.
    - You can also use the `help()` function to see documentation.
- The open function returns a varying type of object depending on how the file was opened.
- https://docs.python.org/3/library/functions.html#open
- Example: Opening a text file returns a subclass of io.TextIOBase:
    - https://docs.python.org/3/library/io.html#io.TextIOBase

NIM

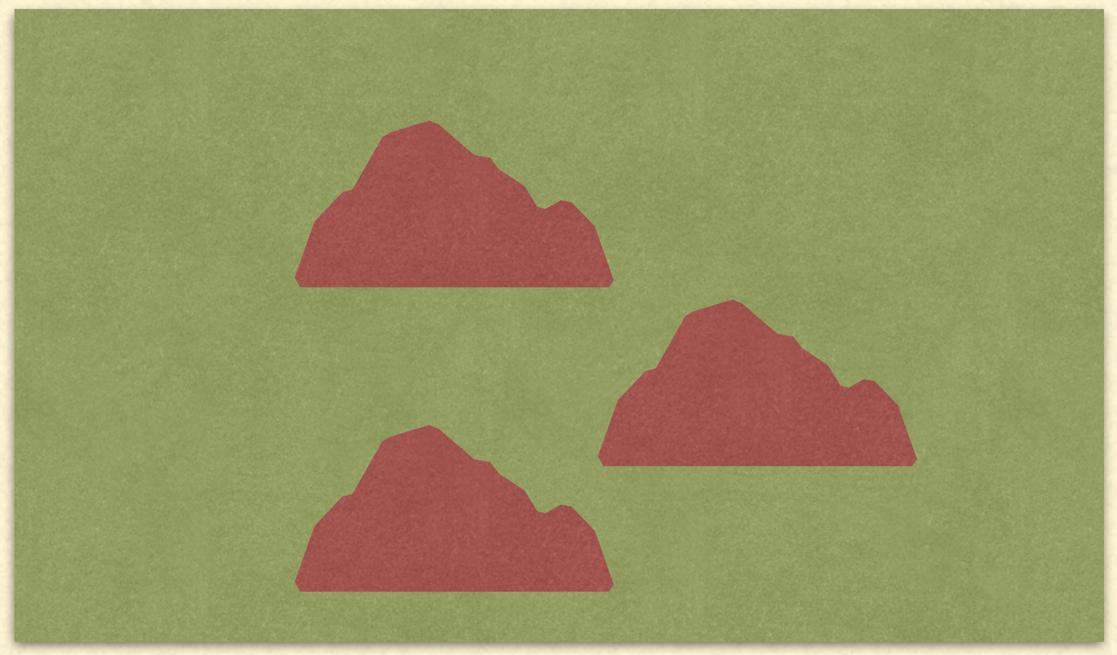
NIM

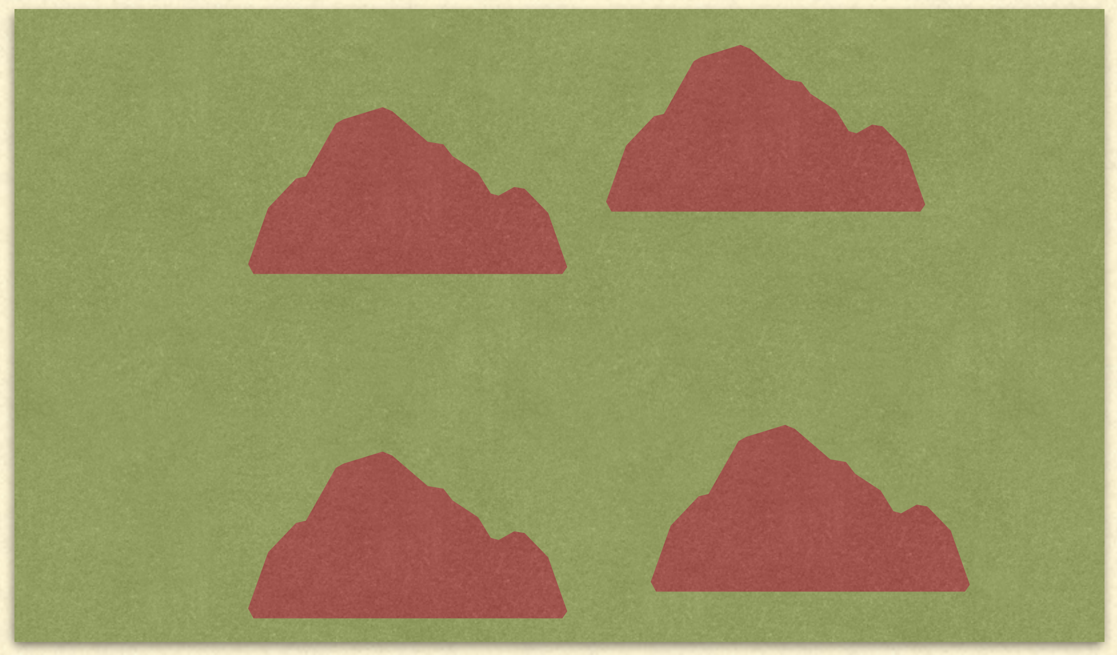
- 3 piles of stones, 2 players
- Players alternate taking stones from a single pile.
  - Each player **must** take stones from one of the piles.
  - Player cannot take stones such that there will be none left in all three piles.
- Game ends when a total of one stone is left.
- Player left with one stone at the start of their turn loses.

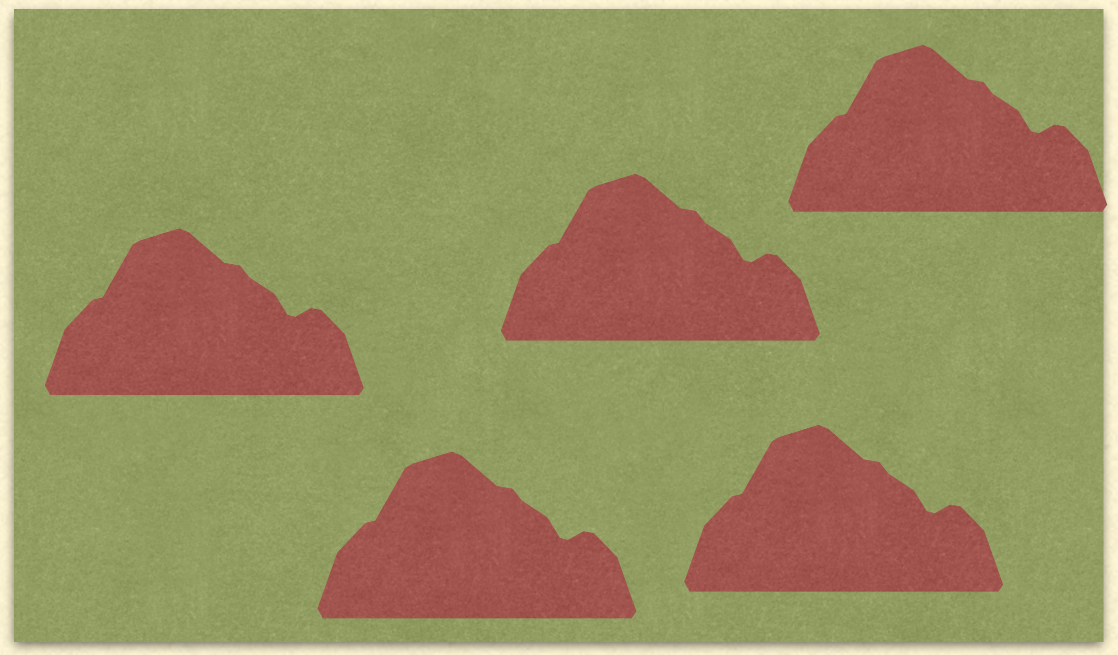
SAMPLE GAME

SAMPLE GAME

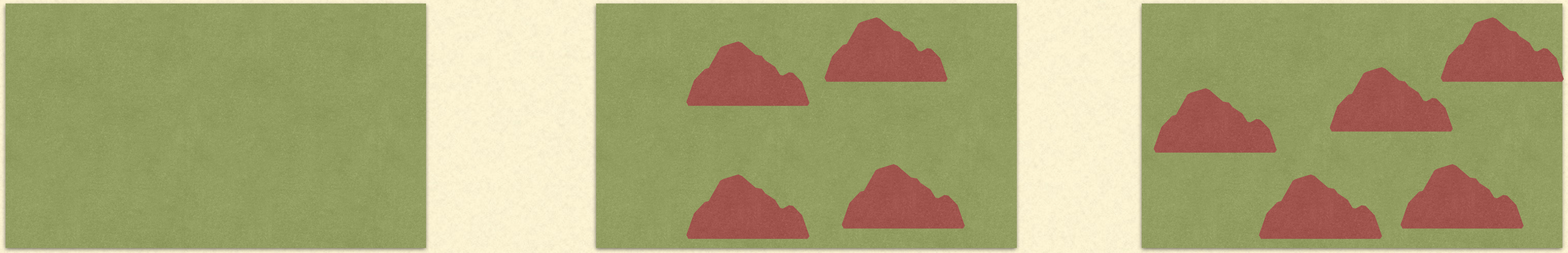






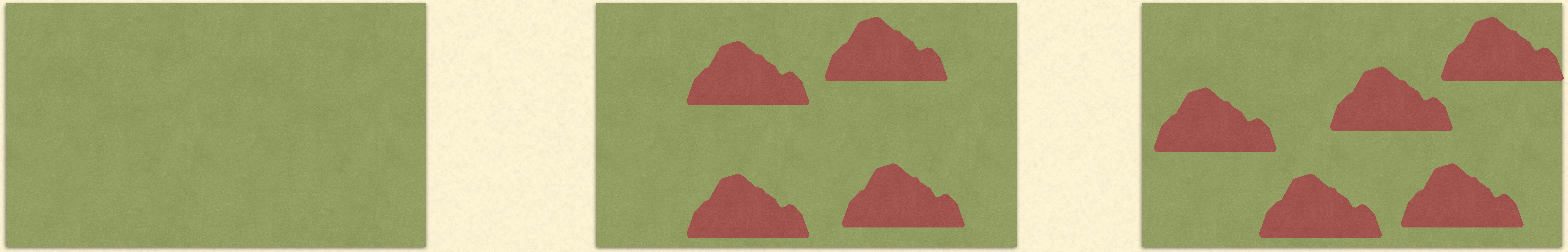
Player 1: remove 3 stones from pile 1

SAMPLE GAME



Player 1: remove 3 stones from pile 1

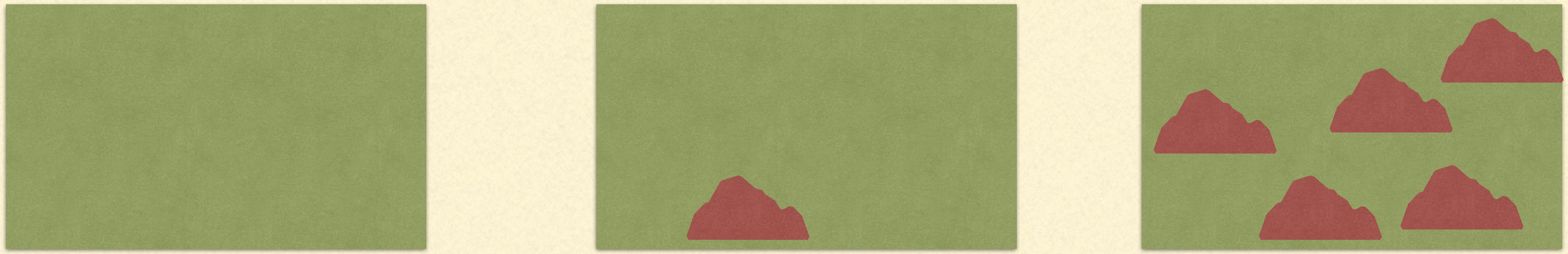
SAMPLE GAME



Player 1: remove 3 stones from pile 1

Player 2: remove 3 stones from pile 2

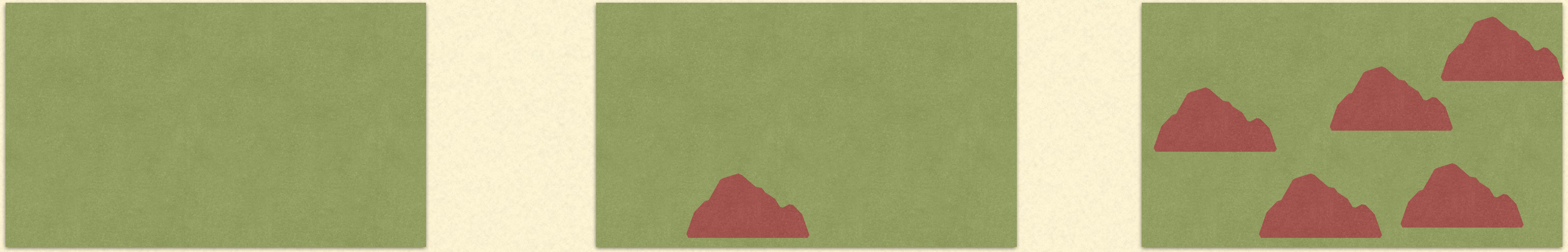
SAMPLE GAME



Player 1: remove 3 stones from pile 1

Player 2: remove 3 stones from pile 2

SAMPLE GAME

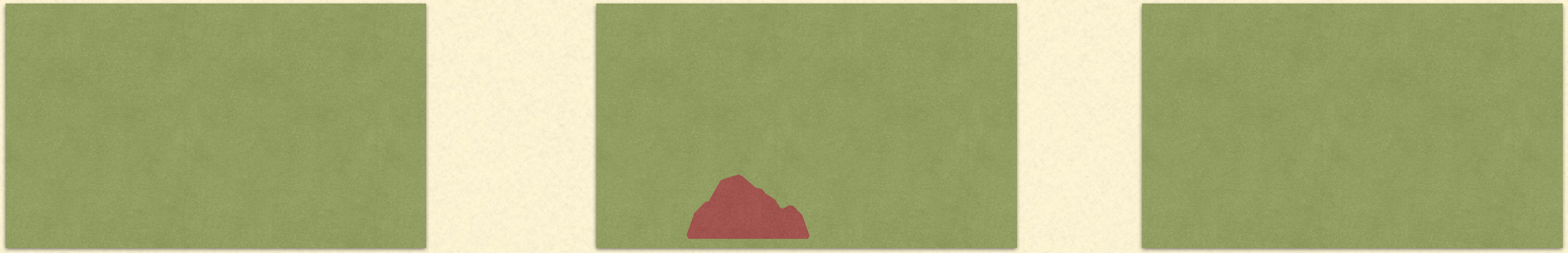


Player 1: remove 3 stones from pile 1

Player 2: remove 3 stones from pile 2

Player 1: remove 5 stones from pile 3

SAMPLE GAME

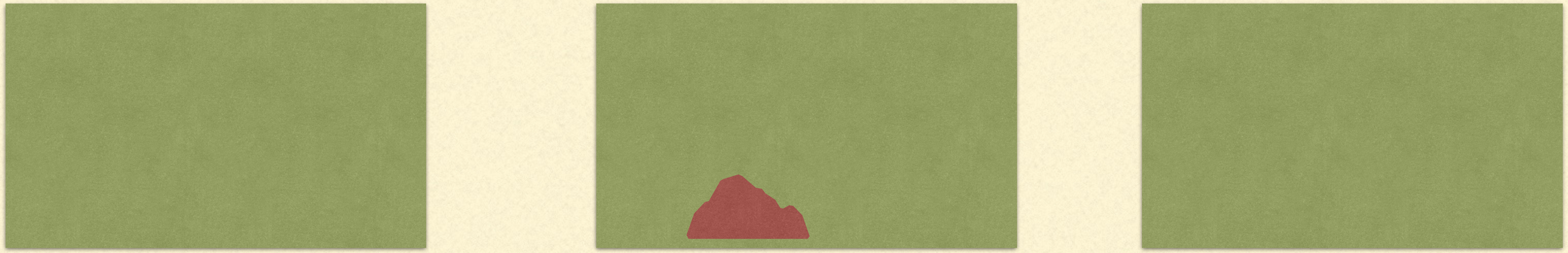


Player 1: remove 3 stones from pile 1

Player 2: remove 3 stones from pile 2

Player 1: remove 5 stones from pile 3

SAMPLE GAME



Player 1: remove 3 stones from pile 1

Player 2: remove 3 stones from pile 2

Player 1: remove 5 stones from pile 3

Player 1 is the winner

DEMO: PYTHON NIM