



KodeKloud

Course Objectives



Syntax, variables, data types, numbers and strings



Conditional Statements (else, if, and elif)



Loops, logic and bit operations, lists, tuples and dictionaries



Solid understanding of Python through hands-on exercises and projects



KodeKloud



Introduction

Easy and intuitive programming language

Free and Open Source

Can be widely used for a variety of tasks





bash

```
$ python3
```



python3

```
$ python3
```

```
Python 3.8.2 (default, Oct 2 2020, 10:45:41)
```

```
[Clang 12.0.0 (clang-1200.0.32.27)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more information
```

```
>>>
```


Python


PSF

Docs

PyPI

Jobs

Community



Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events


Download the latest version for Windows

Download Python 3.7.0

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Pre-releases](#)




Looking for Python 2.7? See below for specific releases



Join the official **Python Developers Survey 2018** and win valuable prizes: [Start the survey!](#)

Looking for a specific release?

Python releases by version number:

| Release version | Release date | Click for more | |
|-----------------|--------------|--|---------------|
| Python 3.5.6 | 2018-08-02 |  Download | Release Notes |
| Python 3.4.9 | 2018-08-02 |  Download | Release Notes |
| Python 3.7.0 | 2018-06-27 |  Download | Release Notes |



python3

```
$ python3
```

```
Python 3.8.2 (default, Oct 2 2020, 10:45:41)
```

```
[Clang 12.0.0 (clang-1200.0.32.27)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more information
```

```
>>>
```



python3

```
$ python3
```

```
Python 3.8.2 (default, Oct 2 2020, 10:45:41)
```

```
[Clang 12.0.0 (clang-1200.0.32.27)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more information
```

```
>>> print ( "Hello future Python programmer!" )
```



python3

```
$ python3
```

```
Python 3.8.2 (default, Oct 2 2020, 10:45:41)
```

```
[Clang 12.0.0 (clang-1200.0.32.27)] on darwin
```

```
Type "help", "copyright", "credits" or "license" for more information
```

```
>>> print ( "Hello future Python programmer!" )
```

```
Hello future Python programmer!
```

myfile.py

```
1 print ( "Hello future Python programmer!" )
```



bash

\$

myfile.py

```
1  print ( "Hello future Python programmer!" )
```



bash

```
$ python myfile.py
```

myfile.py

```
1 print ( "Hello future Python programmer!" )
```



bash

```
$ python myfile.py
```

```
Hello future Python programmer!
```

```
$
```



KodeKloud



Functions - Print()



python3

```
>>> print ( "Hello future Python programmer!" )
```



python3

```
>>> print ( " Hello future Python programmer!      ")  
Hello future Python programmer!
```



python3

```
>>>
```

```
Hello future Python programmer!
```

print

(

"

Hello future Python programmer!

"

)



python3

```
>>>
```

```
Hello future Python programmer!
```

print

(

"

Hello future Python programmer!

"

)



python3

```
>>> print ( " Hello future Python programmer!      ")  
Hello future Python programmer!
```

Functions

A part of your code that's used to cause an effect or evaluate a value.



python3

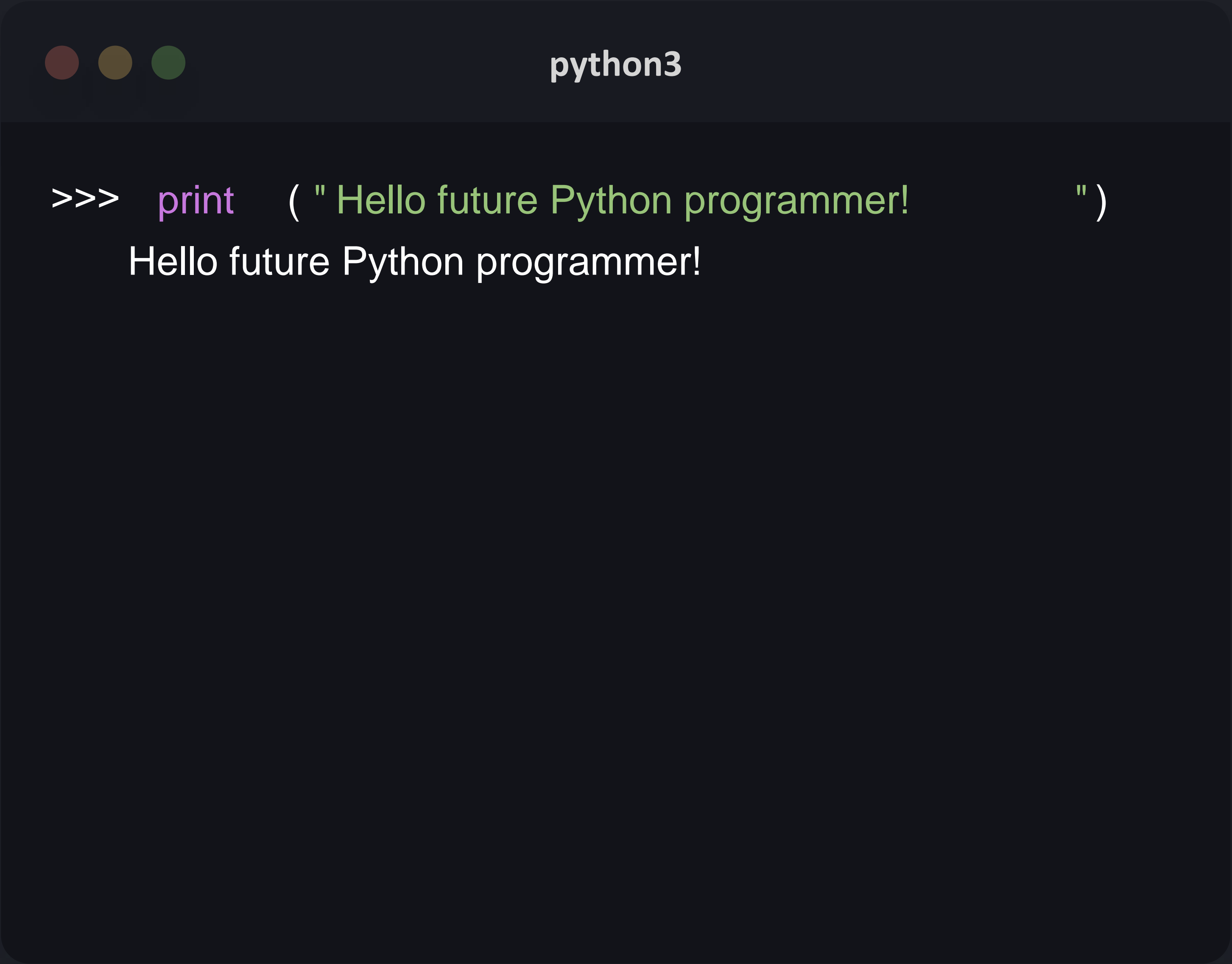
```
>>> print ( " Hello future Python programmer!      ")  
Hello future Python programmer!
```

Functions

A part of your code that's used to cause an effect or evaluate a value.

Can come from:

- Python (built-in functions)
- Modules
- Your own code



```
>>> print ( " Hello future Python programmer! ")  
Hello future Python programmer!
```




python3

```
>>> print ( " Hello future Python programmer!      ")  
Hello future Python programmer!
```



python3

>>>

Hello future Python programmer!

print

(

"

Hello future Python programmer!

"

)



python3

>>>

Hello future Python programmer!

print

(

"

Hello future Python programmer!

"

)



python3

>>>

Hello future Python programmer!

print

(

"

Hello future Python programmer!

"

)



python3

>>>

Hello future Python programmer!

print

(

"

Hello future Python programmer!

"

)



python3

>>>

Hello future Python programmer!

print

(

"

Hello future Python programmer!

"

)



python3

```
>>> print ( " Hello future Python programmer!      ")
```

```
Hello future Python programmer!
```

```
>>> print("Python is a great language")
```

```
Python is a great language
```

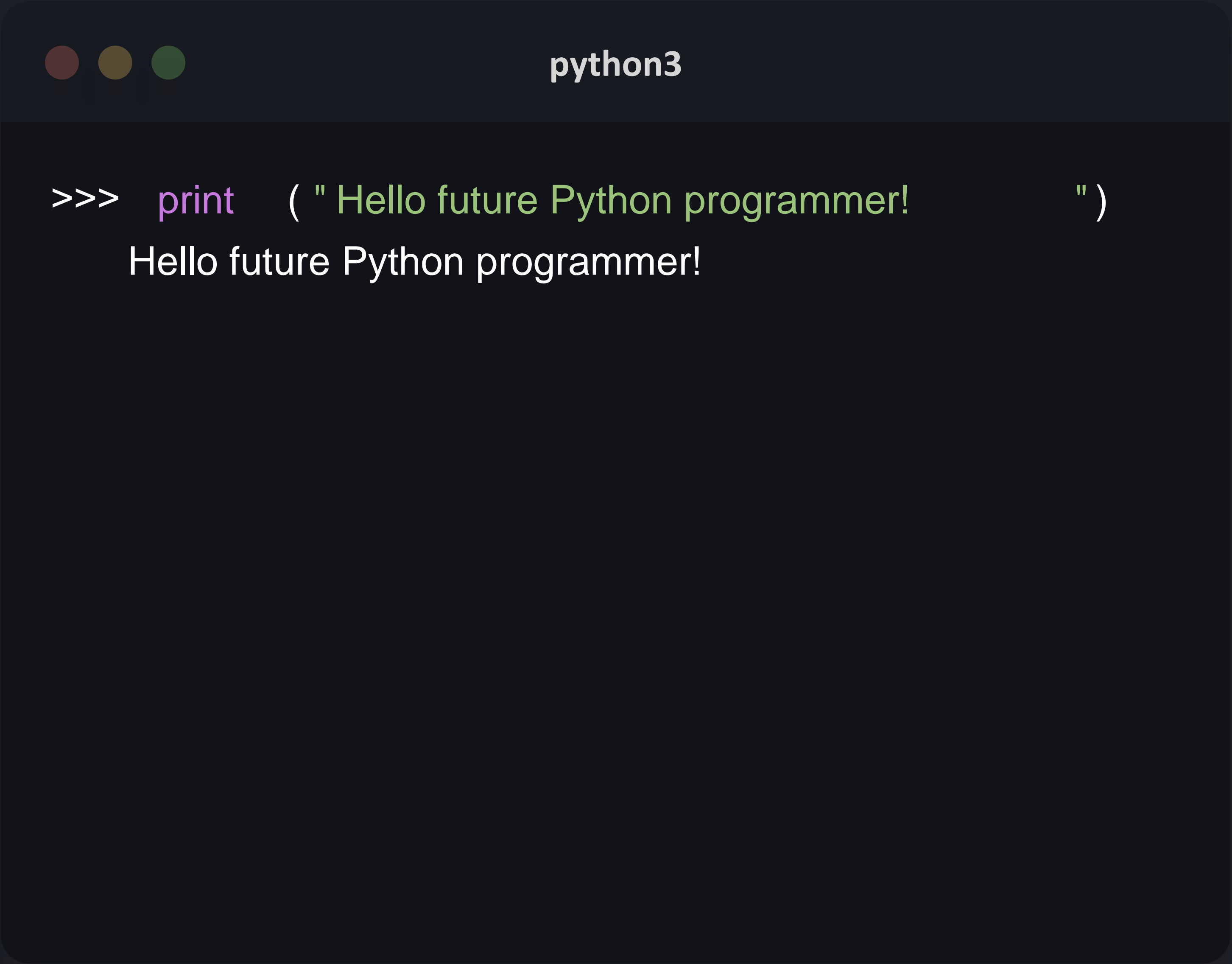
```
>>> print("Strings don't get executed as code")
```

```
Strings don't get executed as code
```

Function Execution

Python:

1. Checks function name
2. Checks arguments passed
3. Jumps into the function
4. Executes the function
5. Returns to your code
6. Resumes execution



```
>>> print ( " Hello future Python programmer! ")  
Hello future Python programmer!
```




python3

```
>>> print ( "Hello future Python programmer!" )
```



python3

```
>>> print ( "Hello future Python programmer!" )  
      print("Python is a great language")  
      print("Strings don't get executed as code")
```



python3

```
>>> print ( "Hello future Python programmer!" )  
      print("Python is a great language")  
      print("Strings don't get executed as code")
```

Hello future Python programmer!

Python is a great language

Strings don't get executed as code



python3

```
>>> print ( "Hello,\nfuture Python programmer!" )
```



python3

```
>>> print ( "Hello,\nfuture Python programmer!" )  
Hello,  
future Python programmer!
```



python3

```
>>> print ( "Hello future Python programmer!" )
```



python3

```
>>> print ( "Hello", "future", "Python", "programmer!" )
```



python3

```
>>> print ( "Hello", "future", "Python", "programmer!" )  
Hello future Python programmer!
```




python3

```
>>> print("Hello!"      )  
      print("Python is a great language")
```



python3

```
>>> print("Hello!"      , end=""    )  
      print("Python is a great language")
```



python3

```
>>> print("Hello!"      , end=""    )  
      print("Python is a great language")
```



python3

```
>>> print("Hello!"      , end="" )  
      print("Python is a great language")  
Hello!Python is a great language
```



python3

```
>>> print("Hello!"      , end="" )  
      print("Python is a great language")  
Hello!Python is a great language  
  
>>> print("Hello!"      , end="!" )  
      print("Python is a great language")  
Hello!!Python is a great language  
  
>>> print("Hello!"      , end="💚" )  
      print("Python is a great language")  
Hello!💚 Python is a great language
```



python3

```
>>> print ( "Hello", "future", "Python", "programmer!" )
```



python3

```
>>> print ( "Hello", "future", "Python", "programmer!" , sep="-" )  
Hello-future-Python-programmer!
```

```
>>> print      "Hello", "future", "Python", "programmer!" , sep="💖" )  
Hello 💖 future 💖 Python 💖 programmer!
```



python3

```
>>> print ( "Hi", "Hello" , sep="! " , end="💖\n" )  
      print ( "So", "enjoying python?" , sep=" , " , end="😊" )  
Hi! Hello 💖  
So, enjoying Python? 😊
```



```
print()
```

- Built-in function: can be used without importing it.
- Allows us to print values to the console
- We can invoke it with parentheses.
- We can pass the value we want to print as arguments between the parentheses.
- The backslash `\` tells python that the next character has a special meaning (eg. `\n`)
- Keyword arguments such as `sep` and `end` can be used to format the output.



KodeKloud



Literals

200

"Hello!"

"Python"

-89

200

"Hello!"

"Python"

-89

name

c

age

print

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

200

1298901

-90

1_000_000

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0o123

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0o123

2

1

0

1

2

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0o123

8^2

1

8^1

2

8^0

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0o123

64

1

8

2

1

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0o123

64

16

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans



0x123

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0x123

2

1

0

1

2

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0x123

16^2

1

16^1

2

16^0

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0x123

256

1

16

2

1

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0x123

256

32

3

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

0x123

291

Literals

Literal types:

1. Integers
 - Octal numbers
 - Hexadecimal numbers
2. Floating point numbers
3. Strings
4. Booleans

45.50

12.1

-90.0

89.394

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

1e-22

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

"Hello!"

'Hello!'

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

```
'Hello! "Python" is cool'
```

```
"Hello! 'Python' is cool"
```

Literals

Literal types:

1. Integers

- Octal numbers
- Hexadecimal numbers

2. Floating point numbers

3. Strings

4. Booleans

```
"Hello! \"Python\" is cool"
```


Literals

Literal types:

1. Integers
 - Octal numbers
 - Hexadecimal numbers
2. Floating point numbers
3. Strings
4. Booleans

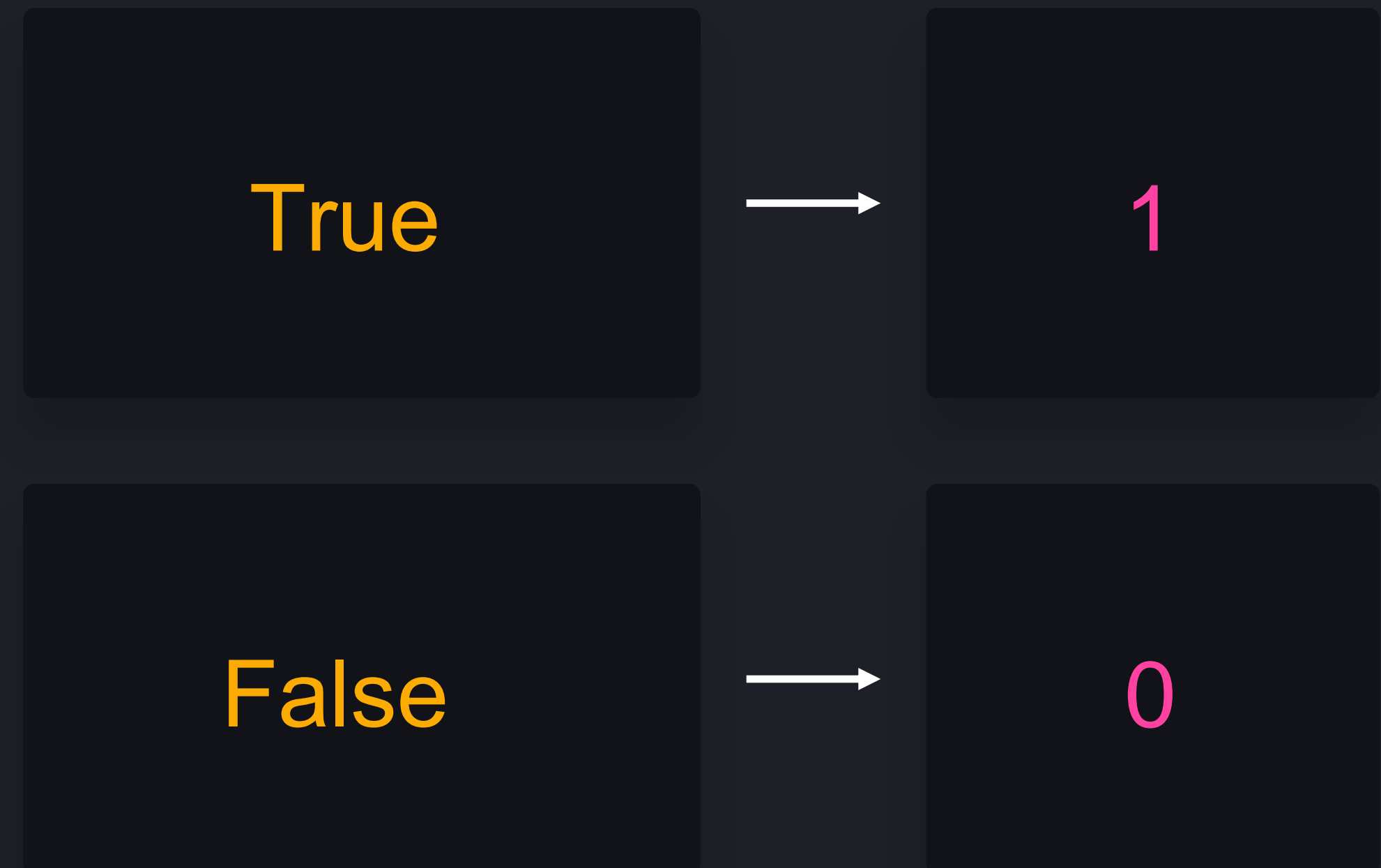
True

False

Literals

Literal types:

1. Integers
 - Octal numbers
 - Hexadecimal numbers
2. Floating point numbers
3. Strings
4. Booleans



Literals

Numbers

- Integers

123

- Octal

0o123

- Hexadecimal

0x123

- Floating point

123.45

Strings

- Double quotes

"Hello!"

- Single quotes

'Hello!'

- Use quotes within strings

'Hi "hi"'

- Escape quotes

'Hi \'hi\''

Boolean

- False

False

- True

True

- Numeric false

0

- Numeric true

1



KodeKloud



Operators

Arithmetic Operators

+

Add

-

Subtract

*

Multiply

/

Divide

//

Floor Divide

%

Modulo

**

Exponential

Arithmetic Operators

Exponential

2³

Arithmetic Operators

Exponential

2 ** 3

Arithmetic Operators

Exponential



python3

```
>>> print(2 ** 3)
```

```
8
```

```
>>> print(2. ** 3.)
```

```
8.0
```

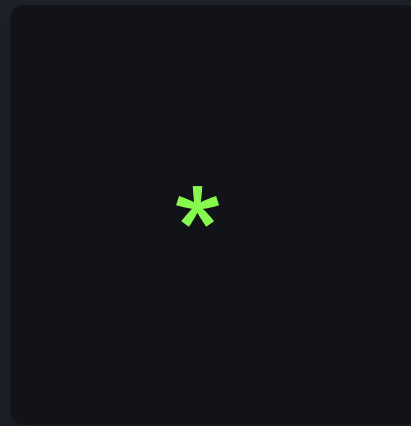
```
>>> print(2 ** 3.)
```

```
8.0
```

```
>>> print(2. ** 3)
```

```
8.0
```

Arithmetic Operators



Multiplication



python3

```
>>> print(2 * 3)
```

```
6
```

```
>>> print(2. * 3.)
```

```
6.0
```

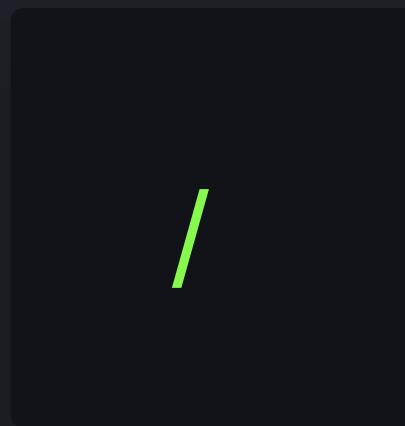
```
>>> print(2 * 3.)
```

```
6.0
```

```
>>> print(2. * 3)
```

```
6.0
```

Arithmetic Operators



Division



python3

```
>>> print(10 / 2)
```

```
5.0
```

```
>>> print(10. / 2.)
```

```
5.0
```

```
>>> print(10 / 2.)
```

```
5.0
```

```
>>> print(10. / 2)
```

```
5.0
```

Arithmetic Operators



//

Floor Division

```
>>> print(10 // 2)
```

```
5
```

```
>>> print(10. // 2.)
```

```
5.0
```

```
>>> print(10 // 2.)
```

```
5.0
```

```
>>> print(10. // 2)
```

```
5.0
```

Arithmetic Operators



//

Floor Division



python3

```
>>> print(6. / 4)
```

```
1.5
```

```
>>> print(6. // 4)
```

```
1.0
```

```
>>> print(6. / -4)
```

```
-1.5
```

```
>>> print(6. // -4)
```

```
-2.0
```

Arithmetic Operators

%

Modulo



python3

```
>>> print(4 % 2)
```

Arithmetic Operators

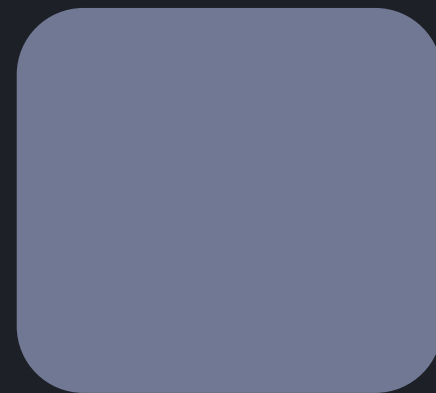
%

Modulo



python3

```
>>> print(4 % 2)
```



Arithmetic Operators

%

Modulo



python3

```
>>> print(4 % 2)
```



Arithmetic Operators

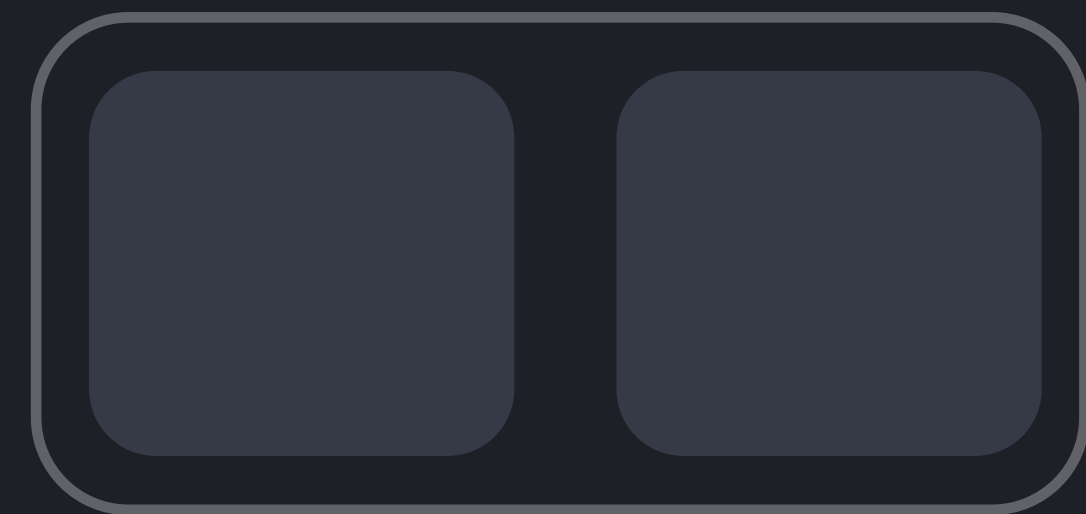
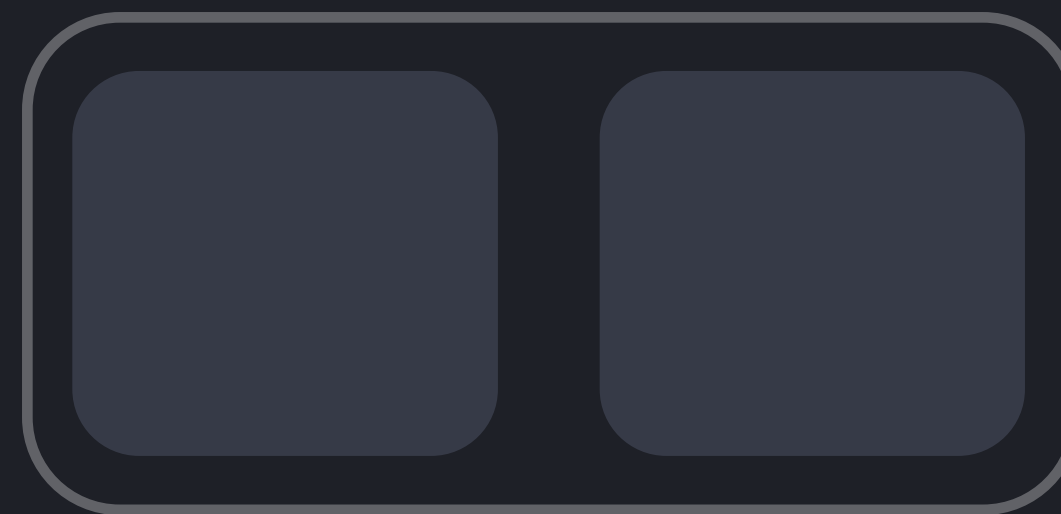
%

Modulo



python3

```
>>> print(4 % 2)  
0
```



Arithmetic Operators

%

Modulo

python3

```
>>> print(5 % 2)
```



Arithmetic Operators

%

Modulo

python3

```
>>> print(5 % 2)
```



Arithmetic Operators

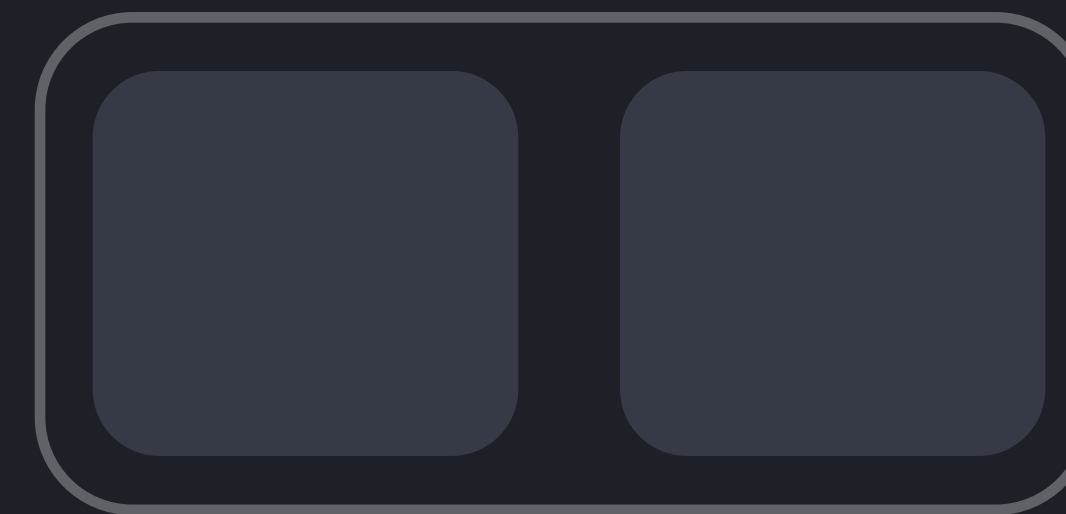
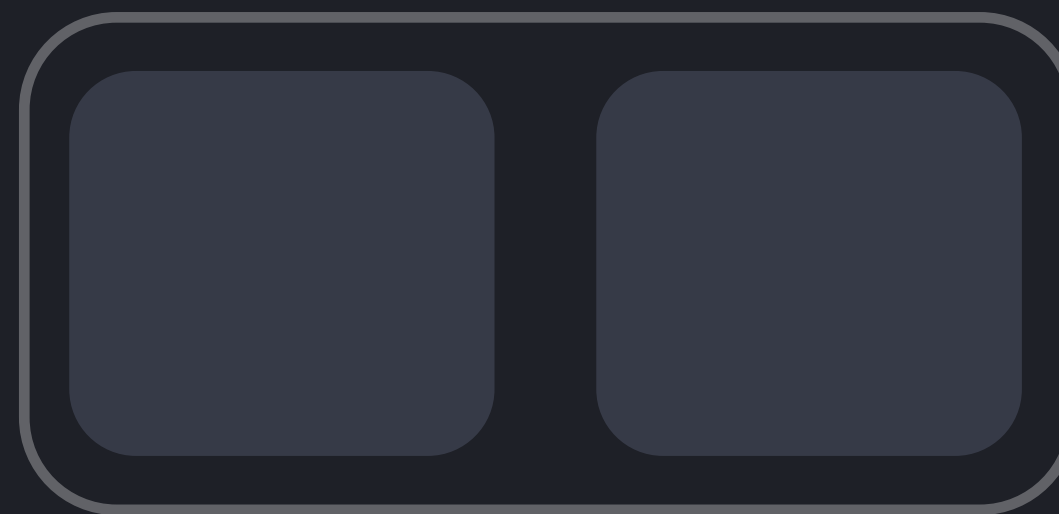
%

Modulo



python3

```
>>> print(5 % 2)
```



Arithmetic Operators

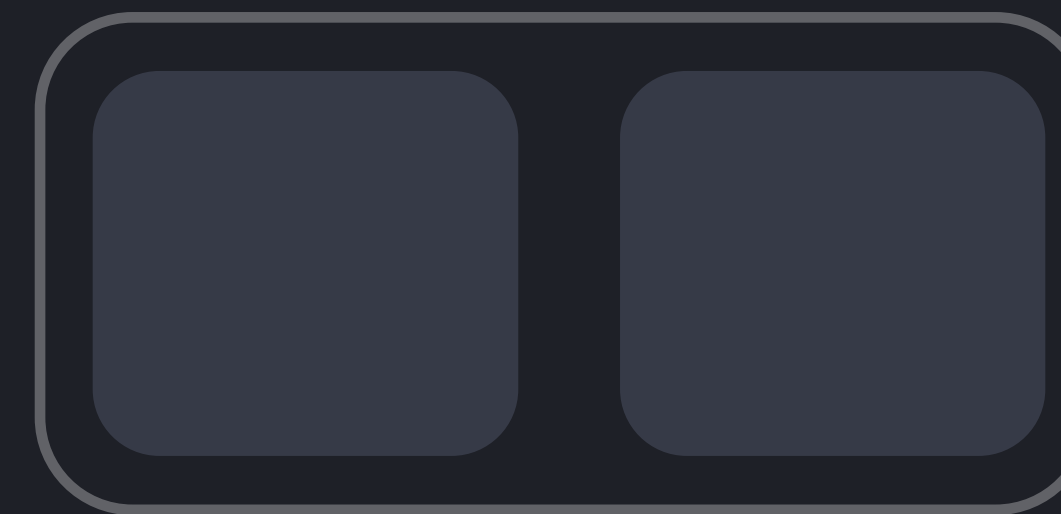
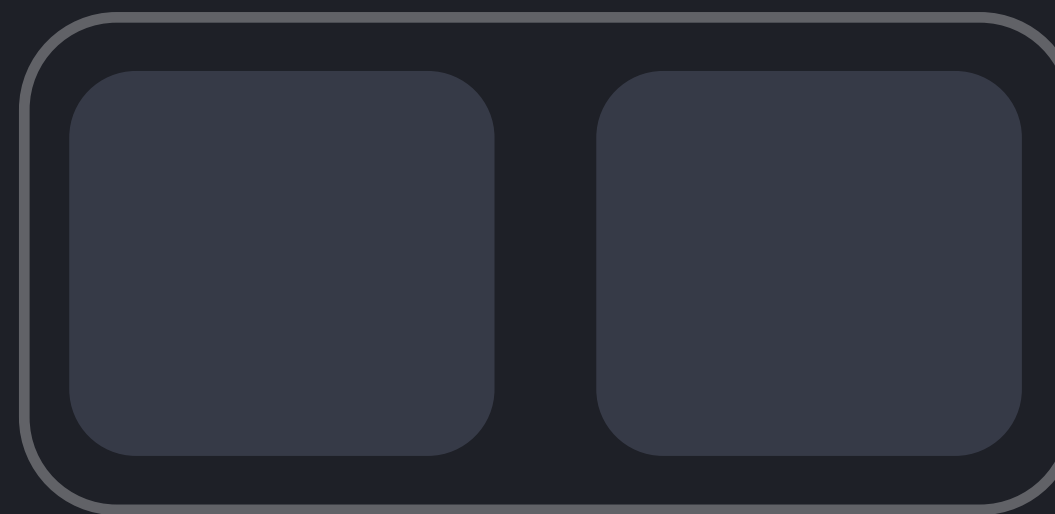
%

Modulo



python3

```
>>> print(5 % 2)  
1
```



Arithmetic Operators

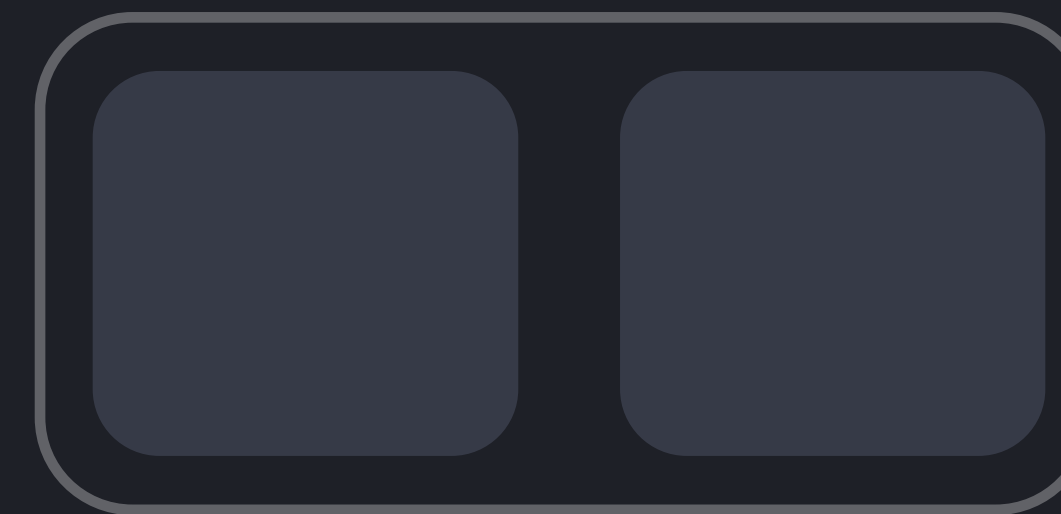
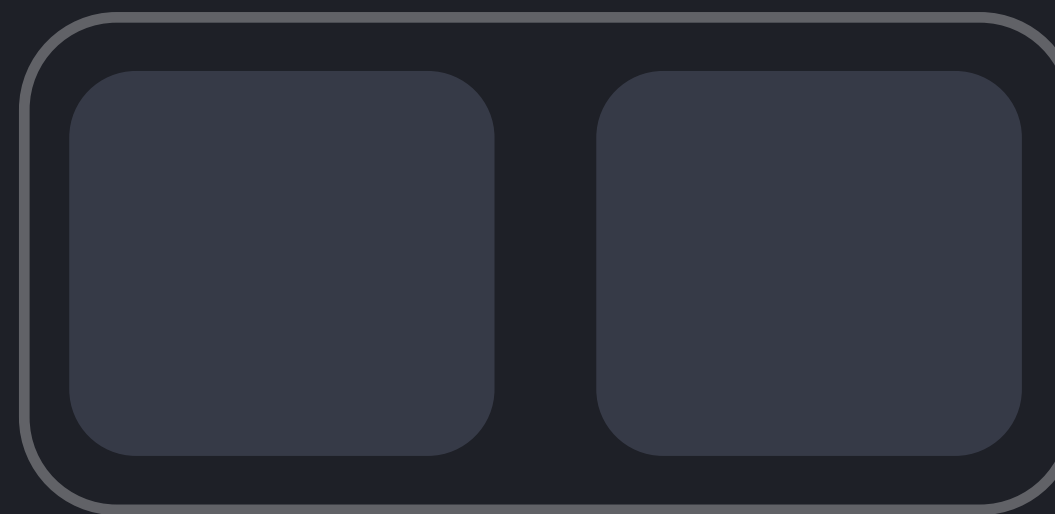
%

Modulo

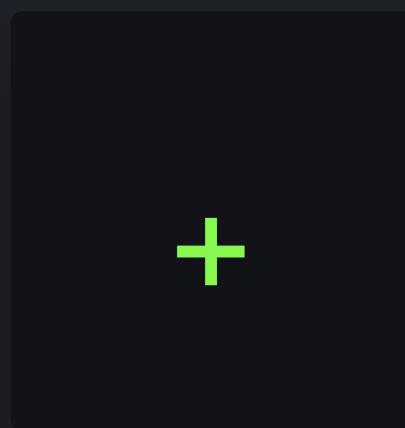


python3

```
>>> print(5 % 2)  
1
```



Arithmetic Operators



Addition



python3

```
>>> print(6 + 4)
```

```
10
```

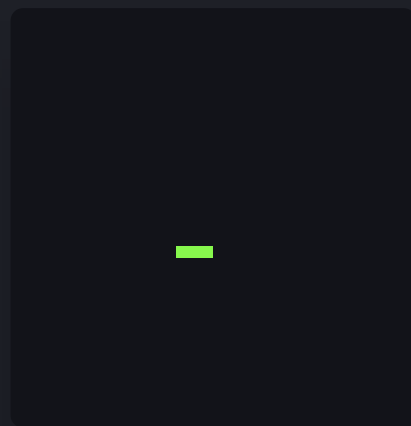
```
>>> print(6. + 4)
```

```
10.0
```

```
>>> print(6. + 4.)
```

```
10.0
```

Arithmetic Operators



Subtraction



python3

```
>>> print(6 - 4)
```

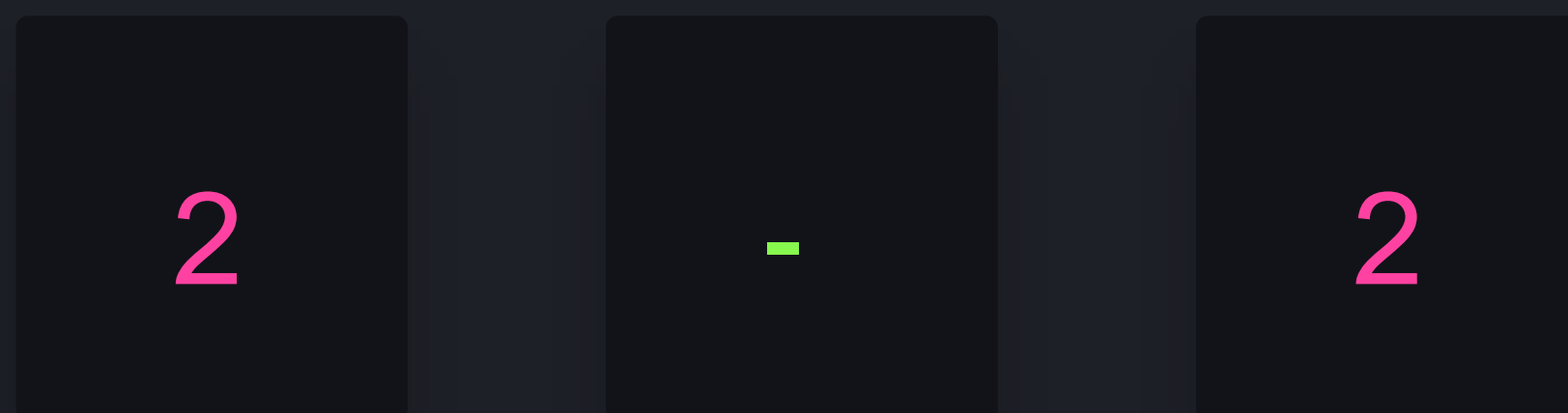
```
2
```

```
>>> print(6. - 4)
```

```
2.0
```

```
>>> print(6. - 4.)
```

```
2.0
```

Binary Operator

$$2 - 2$$

Binary Operator

$$2 - 2$$

Binary Operator

2

-

2

Binary Operator

-

2

Unary Operator



python3

```
>>> print(-6 - 6)
```



python3

```
>>> print(-6 - 6)
```

```
-12
```



python3

```
>>> print(-6 - 6)
```

```
-12
```

```
>>> print(10 - -6)
```



python3

```
>>> print(-6 - 6)
```

```
-12
```

```
>>> print(10 - -6)
```

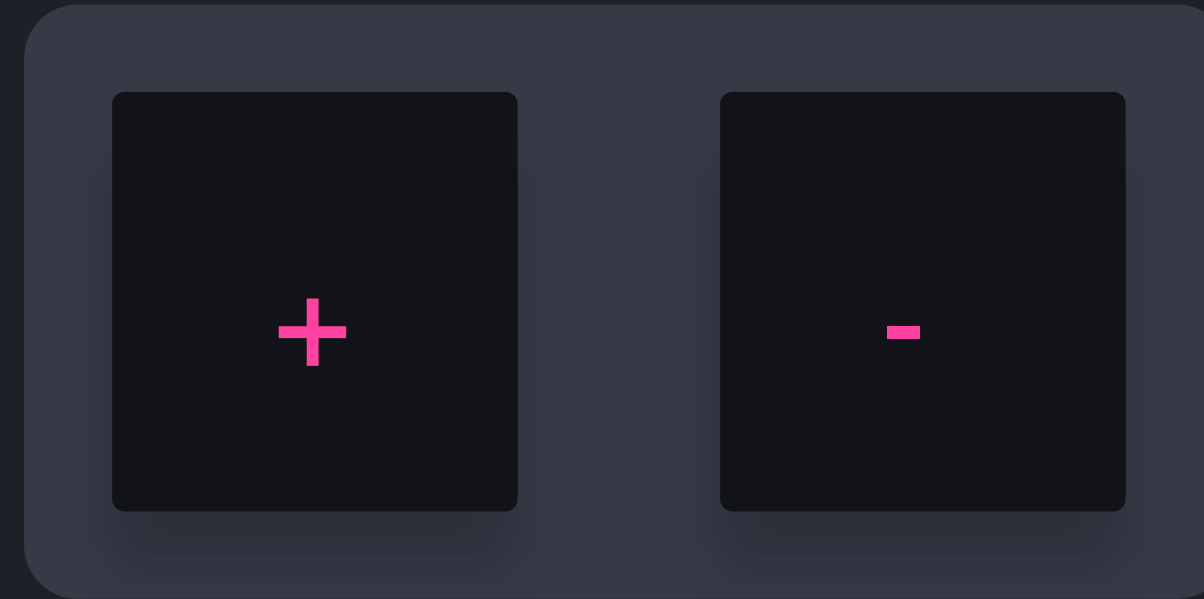
```
16
```



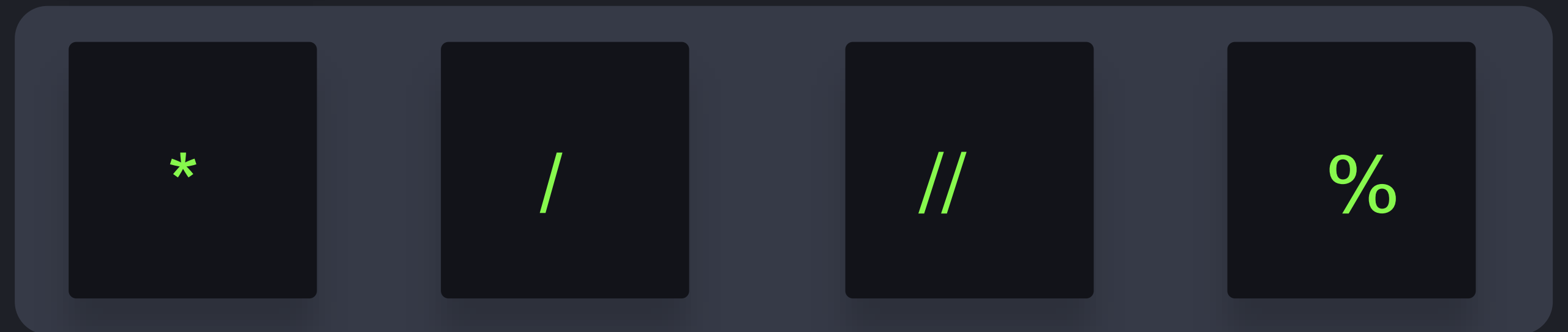

python3

```
>>> print(10 - 6 ** 7 / 9 * 23 + 1)
```

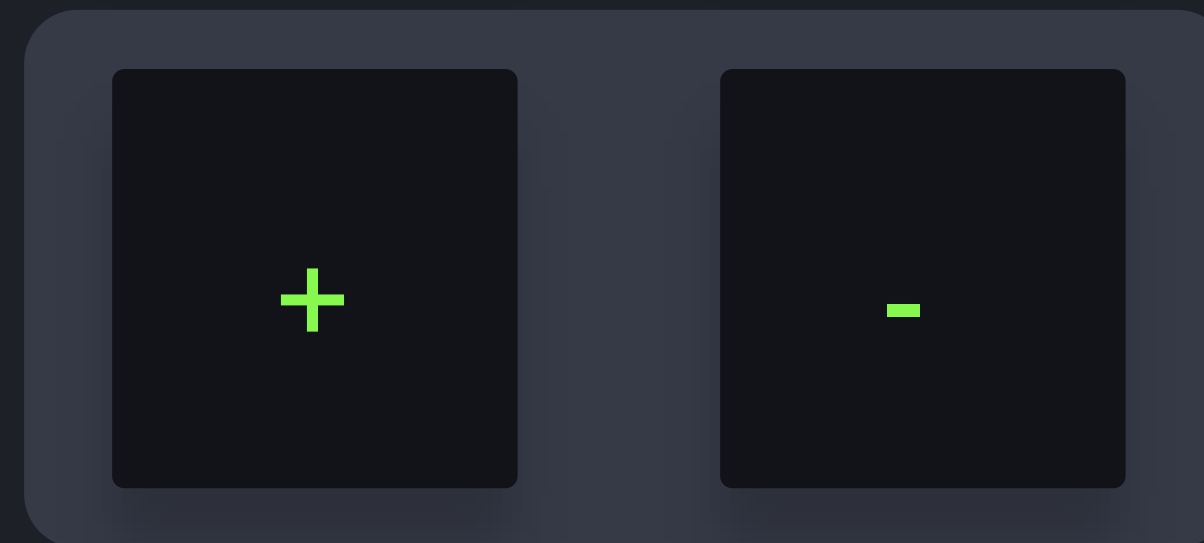
Highest Priority



(unary)



Lowest Priority



(binary)



python3

```
>>> print(10 - 6 ** 2 / 9 * 10 + 1)
```

```
print( 10 - 6 ** 2 / 9 * 10 + 1)
```

```
print( 10 - 6 ** 2 / 9 * 10 + 1 )
```

```
print( 10 - 36 / 9 * 10 + 1 )
```

```
print( 10 - 4 * 10 + 1 )
```

```
print( 10 - 40 + 1 )
```



```
print( 10 - 40 + 1 )
```

```
print( -30 + 1 )
```

```
print( -29 )
```



python3

```
>>> print(10 - 6 ** 7 / 9 * 10 + 1)
```

```
-29
```



python3

```
>>> print(2 * (2 + 3) )
```



python3

```
>>> print(2 * 5)
```



python3

```
>>> print(2 * 5)  
10
```



KodeKloud



Variables



python3

```
>>> print(2 * 5)
```

```
10
```



python3

```
>>> amount_of_apples = 2
```

```
>>> cost_of_apple = 5
```



python3

```
>>> amount_of_apples = 2
>>> cost_of_apple = 5
>>> print(amount_of_apples * cost_of_apple)
```



python3

```
>>> amount_of_apples = 2
>>> cost_of_apple = 5
>>> print(amount_of_apples * cost_of_apple)
10
```

amount_of_apples

2

cost_of_apple

5

Valid Variable Names

amount_of_apples
cost_of_apple
_total_cost

Invalid Variable Names

am*unt_o%_apples
c*st_o%_apple
5apples_cost

Valid Variable Names

amount_of_apples

cost_of_apple

_total_cost

COST_OF_APPLE

Invalid Variable Names

am*unt_o%_apples

c*st_o%_apple

5apples_cost

Valid Variable Names

amount_of_apples
cost_of_apple
_total_cost
COST_OF_APPLE

Invalid Variable Names

am*unt_o%_apples
c*st_o%_apple
5apples_cost
del
elif
return

Valid Variable Names

amount_of_apples
cost_of_apple
_total_cost
COST_OF_APPLE

Invalid Variable Names

am*unt_o%_apples
c*st_o%_apple
5apples_cost
del
elif
return

Reserved Keywords

False

None

True

and

as

assert

break

class

continue

def

del

elif

else

except

finally

for

from

global

if

import

in

is

lambda

nonlocal

not

or

pass

raise

return

try

while

with

yield

Valid Variable Names

Import

Del

Elif

Return

Invalid Variable Names

import

del

elif

return

amount_of_apples

2

cost_of_apple

5



python3

```
>>> cost_of_apple = cost_of_apple + 2
```

amount_of_apples

2

cost_of_apple

7



python3

```
>>> cost_of_apple = cost_of_apple + 2
```

amount_of_apples

2

cost_of_apple

7



python3

```
>>> cost_of_apple = cost_of_apple + 2
>>> print(amount_of_apples * cost_of_apple)
```

amount_of_apples

2

cost_of_apple

7



python3

```
>>> cost_of_apple += cost_of_apple + 2
```

```
>>> print(amount_of_apples * cost_of_apple)
```

18

amount_of_apples

2

cost_of_apple

5

```
python3
cost_of_apple = cost_of_apple + 2
>>>
>>> print(amount_of_apples * cost_of_apple)
14
```

amount_of_apples

2

cost_of_apple

5



python3
cost_of_apple

+= 2

>>>

>>> print(amount_of_apples * cost_of_apple)

14

amount_of_apples

2

cost_of_apple

5



python3

```
>>> cost_of_apple += 2
```

amount_of_apples

2

cost_of_apple

7



python3

```
>>> cost_of_apple += 2
>>> print(amount_of_apples * cost_of_apple)
```

amount_of_apples

2

cost_of_apple

7



python3

```
>>> cost_of_apple += 2
>>> print(amount_of_apples * cost_of_apple)
18
```

Without Shortcut Operator

```
cost_of_apple = cost_of_apple + 2
```

```
cost_of_apple = cost_of_apple - 2
```

```
cost_of_apple = cost_of_apple * 2
```

```
cost_of_apple = cost_of_apple ** 2
```

```
cost_of_apple = cost_of_apple / 2
```

```
cost_of_apple = cost_of_apple // 2
```

```
cost_of_apple = cost_of_apple % 2
```

With Shortcut Operator

```
cost_of_apple += 2
```

```
cost_of_apple -= 2
```

```
cost_of_apple *= 2
```

```
cost_of_apple **= 2
```

```
cost_of_apple /= 2
```

```
cost_of_apple //= 2
```

```
cost_of_apple %= 2
```

Variables

- Variables allow you to store values
- A variable has a valid name (letters, digits, underscore, not a reserved keyword)
- Python is dynamically typed: variables can be redeclared
- We can use shortcut operators in order to cleanly redeclare a variable
- We can combine text and variables using the `+` operator in the `print` function



python3

```
>>> print("One apple costs: " + cost_of_apple)
One apple costs: 5
```



KodeKloud



Comments



python3

```
>>> amount_of_apples = 2           # Amount in basket  
# The cost of an apple in USD  
>>> cost_of_apple = 5
```



python3

```
>>> amount_of_apples = 2           # Amount in basket
```

```
    # The cost of an apple in USD  
    # Should always be an integer
```

```
>>> cost_of_apple = 5
```



python3

```
>>> amount_of_apples = 2           # Amount in basket
```

```
    # The cost of an apple in USD  
    # Should always be an integer
```

```
>>> cost_of_apple = 5
```



python3

```
>>> amount_of_apples = 2
```

```
>>> cost_of_apple = 5
```



python3

```
>>> amount_of_apples = 2
>>> # cost_of_apple = 5
>>> print(amount_of_apples * cost_of_apple)
```



python3

```
>>> amount_of_apples = 2
```

```
>>> # cost_of_apple = 5
```

```
>>> print(amount_of_apples * cost_of_apple)
```

```
NameError: name 'cost_of_apple' is not defined
```



KodeKloud



Input



python3

```
>>> print("Hello!")
```

```
Hello!
```

```
input()
```



python3

```
>>> input("How are you feeling today? ")
```



python3

```
>>> input("How are you feeling today? ")  
How are you feeling today?           Fantastic!
```



python3

```
>>> input("How are you feeling today? ")
```

How are you feeling today? Fantastic!

```
>>>
```



python3

```
>>> favorite_color = input("What is your favorite color? ")
```



python3

```
>>> favorite_color = input("What is your favorite color? ")  
What is your favorite color? blue
```




python3

```
>>> favorite_color = input("What is your favorite color? ")
```

What is your favorite color? blue

```
>>> print("Your favorite color is " + favorite_color)
```



python3

```
>>> favorite_color = input("What is your favorite color? ")
```

What is your favorite color? blue

```
>>> print("Your favorite color is " + favorite_color)
```

Your favorite color is blue



python3

```
>>> age = input("How old are you? ")
```

```
How old are you? 22
```



python3

```
>>> age = input("How old are you? ")
```

```
How old are you?      22
```

```
>>> print(age - 10)
```

```
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```



python3

```
>>> age = input("How old are you? ")
```

```
How old are you? 22
```

Type Casting

- Integers
- Floating point

`int()`

`float()`



python3

```
>>> age = input("How old are you? ")
```

```
How old are you? 22
```

```
>>> print(int(age) - 10)
```

Type Casting

- Integers
- Floating point

`int()`

`float()`



python3

```
>>> age = input("How old are you? ")
```

```
How old are you? 22
```

```
>>> print(int(age) - 10)
```

```
12
```

Type Casting

- Integers
- Floating point

`int()`

`float()`



python3

```
>>> age = int(input("How old are you? "))
```

```
How old are you? 22
```

```
>>> print(age - 10)
```

```
12
```


`input()`

- Prompts the user to input some data from the console
- It accepts an optional parameter that can be used in order to write a message before the user input
- Always returns a string
- A program that doesn't use any input function, is called a **deaf program**



KodeKloud



String Methods

+

*



python3

```
>>> print(10 + 2)
```

```
12
```

```
>>> print(10 * 2)
```

```
22
```

+



python3

```
>>> print(10 + 2)
```

```
12
```

```
>>> print("Hello" + " " + "there!")
```

```
"Hello there!"
```

*



python3

```
>>> print(10 * 2)
```

```
22
```

*



python3

```
>>> print(10 * 2)
```

```
22
```

```
>>> print("ha" * 10)
```

```
"hahahahahahahahaha"
```

*



python3

```
>>> print("ha" * 10)
"hahahahahahahahaha"
```

```
>>> print("ha" * 2)
"haha"
```

```
>>> print("ha" * 0)
""
```

```
>>> print("ha" * -1)
""
```




python3

```
>>> print(int("22"))
```

```
22
```



python3

```
>>> print(int("22"))
```

```
22
```

```
>>> print(str(22))
```

```
"22"
```



python3

```
>>> cost_of_apple = 2
```

```
>>> amount_of_apples = input("How many apples do you want? ")
```

```
How many apples do you want? 10
```



python3

```
>>> cost_of_apple = 2
```

```
>>> amount_of_apples = input("How many apples do you want? ")
```

```
How many apples do you want? 10
```



python3

```
>>> cost_of_apple = 2
>>> amount_of_apples = input("How many apples do you want? ")
How many apples do you want? 10
>>> total_sum = cost_of_apple * int(amount_of_apples)
```



python3

```
>>> cost_of_apple = 2
>>> amount_of_apples = input("How many apples do you want? ")
How many apples do you want? 10
>>> total_sum = cost_of_apple * int(amount_of_apples)
>>> print("You have to pay: " + str(total_sum))
```



python3

```
>>> cost_of_apple = 2
```

```
>>> amount_of_apples = input("How many apples do you want? ")
```

How many apples do you want? 10

```
>>> total_sum = cost_of_apple * int(amount_of_apples)
```

```
>>> print("You have to pay: " + str(total_sum))
```

You have to pay: 20

String Operations

- You can use `+` in order to concatenate two strings
- You can use `*` in order to repeat a string a several amount of times.
- With the `str` function, you can type-cast a number into a string



KodeKloud



Comparison Operators

Comparison Operators

==

!=

>

>=

<

<=

Comparison Operators

==

Equal



python3

```
>>> print(2 == 2)
```

```
True
```

```
>>> print(2 == 4)
```

```
False
```

```
>>> print("Hello!" == "Hello!")
```

```
True
```

```
>>> print("Hello!" == "Goodbye!")
```

```
False
```

```
>>> print(4 == (2 * 2))
```

```
True
```

Comparison Operators

!=

Not Equal



python3

```
>>> print(2 != 2)
```

```
False
```

```
>>> print(2 != 4)
```

```
True
```

```
>>> print("Hello!" != "Hello!")
```

```
False
```

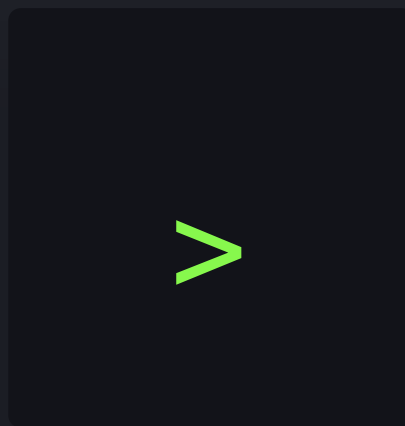
```
>>> print("Hello!" != "Goodbye!")
```

```
True
```

```
>>> print(4 != (2 * 2))
```

```
False
```

Comparison Operators



Greater than



python3

```
>>> print(4 > 2)
```

```
True
```

```
>>> print(2 > 4)
```

```
False
```

```
>>> print(2 > 2)
```

```
False
```

```
>>> cost_of_apple = 2
```

```
>>> cost_of_banana = 3
```

```
>>> print(cost_of_apple > cost_of_banana)
```

```
False
```

Comparison Operators

>=

Greater than or equal to



python3

```
>>> print(4 >= 2)
```

```
True
```

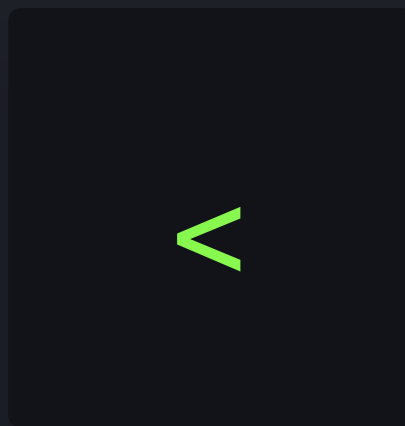
```
>>> print(2 >= 4)
```

```
False
```

```
>>> print(2 >= 2)
```

```
True
```

Comparison Operators



Smaller than



python3

```
>>> print(4 < 2)
```

```
False
```

```
>>> print(2 < 4)
```

```
True
```

```
>>> print(2 < 2)
```

```
False
```

```
>>> cost_of_apple = 2
```

```
>>> cost_of_banana = 3
```

```
>>> print(cost_of_apple < cost_of_banana)
```

```
True
```


Comparison Operators



Smaller than or equal to



python3

```
>>> print(4 <= 2)
```

```
False
```

```
>>> print(2 <= 4)
```

```
True
```

```
>>> print(2 <= 2)
```

```
True
```

Comparison Operators

==

!=

>

>=

<

<=



KodeKloud



Conditional Statements

if *condition*:

```
if condition :  
    print("The condition is true!")
```

If

True

:

`print("The condition is true!")`

```
if False :  
    print("The condition is true!")
```




python3

```
>>> age = int(input("How old are you? "))
```



python3

```
>>> age = int(input("How old are you? "))
```

```
How old are you? 22
```



python3

```
>>> age = int(input("How old are you? "))
```

```
How old are you?      22
```

```
>>> if age >= 18:
```

```
    print("You are an adult!")
```



python3

```
>>> age = int(input("How old are you? "))
```

```
How old are you? 22
```

```
>>> if age >= 18:
```

```
    print("You are an adult!")
```

```
You are an adult!
```

```
if condition :  
    print("The condition is true!")
```

```
if condition:  
    print("The condition is true!")
```

```
if condition :  
    print("The condition is true!")  
else:  
    print("The condition is false!")
```

if

True

:

```
print("The condition is true!")
```

else:

```
print("The condition is false!")
```



```
if False:
```

```
    print("The condition is true!")
```

```
else:
```

```
    print("The condition is false!")
```

```
if condition:  
    print("The condition is true!")  
elif second_condition :  
    print("Only the second condition is true!")  
else:  
    print("Both conditions are false!")
```

```
if False:  
    print("The condition is true!")  
elif True:  
    print("Only the second condition is true!")  
else:  
    print("Both conditions are false!")
```

```
if True:  
    print("The condition is true!")  
elif True:  
    print("Only the second condition is true!")  
else:  
    print("Both conditions are false!")
```

```
if age >= 18:  
    if age == 18:  
        print("You are exactly 18 years old!")  
    else:  
        print("You older than 18 years old!")
```

```
if age >= 18:  
    if age == 18:  
        print("You are exactly 18 years old!")  
    else:  
        print("You older than 18 years old!")
```

```
if age >= 18:  
    if age == 18:  
        print("You are exactly 18 years old!")  
    else:  
        print("You older than 18 years old!")
```



KodeKloud



Loops - While

`while` *condition*:

`while` *condition*:



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
    guess = int(input("Guess a number: "))
```



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
        guess = int(input("Guess a number: "))
Guess a number: 0
```



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
    guess = int(input("Guess a number: "))
Guess a number: 0
Guess a number: 4
```



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
        guess = int(input("Guess a number: "))
```

Guess a number: 0

Guess a number: 4

Guess a number: 3



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
    guess = int(input("Guess a number: "))
Guess a number: 0
Guess a number: 4
Guess a number: 3
>>>
```




python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
    guess = int(input("Guess a number: "))
```



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
        guess = int(input("Guess a number: "))
    else:
        print("Congratulations, you got it!")
```



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
        guess = int(input("Guess a number: "))
    else:
        print("Congratulations, you got it!")
```

Guess a number: 0



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
        guess = int(input("Guess a number: "))
    else:
        print("Congratulations, you got it!")
```

Guess a number: 0

Guess a number: 4



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
        guess = int(input("Guess a number: "))
    else:
        print("Congratulations, you got it!")
```

Guess a number: 0

Guess a number: 4

Guess a number: 3



python3

```
>>> secret_number = 3
>>> guess = int(input("Guess a number: "))
>>> while guess != secret_number:
        guess = int(input("Guess a number: "))
    else:
        print("Congratulations, you got it!")
```

Guess a number: 0

Guess a number: 4

Guess a number: 3

Congratulations, you got it!

```
>>>
```



KodeKloud



Loops - For


```
for ... in ...:
```

```
for i in range(9):
```

```
for i in range(9):
```

0

1

2

3

4

5

6

7

8

9

```
for i in range(9):
```

0

1

2

3

4

5

6

7

8

9

i

0

1

2

3

4

5

6

7

8

9



python3

```
>>> for i in range(9):  
        print("i is: ", i)
```

i is: 0

i

0

1

2

3

4

5

6

7

8

9

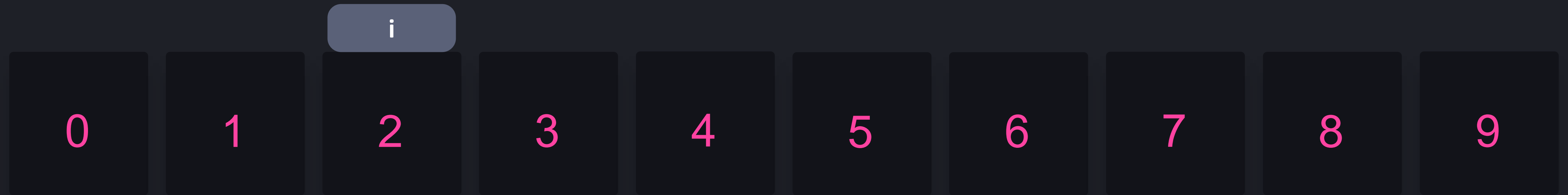


python3

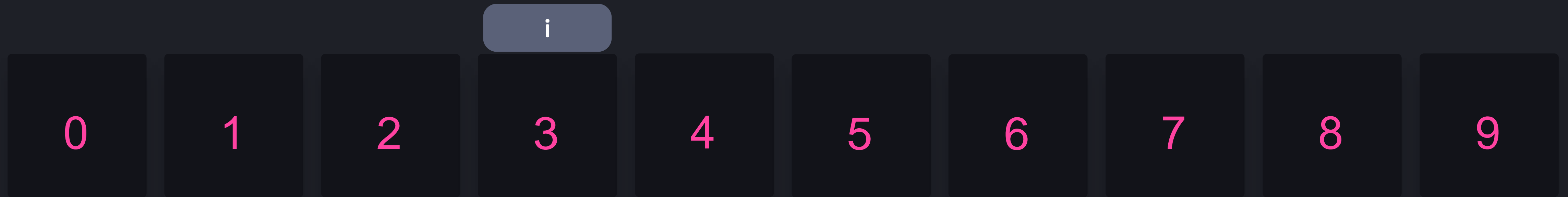
```
>>> for i in range(9):  
        print("i is: ", i)
```

i is: 0

i is: 1



```
>>> for i in range(9):  
        print("i is: ", i)  
  
i is: 0  
i is: 1  
i is: 2
```



```
python3

>>> for i in range(9):
        print("i is: ", i)

i is: 0
i is: 1
i is: 2
i is: 3
```


i

0

1

2

3

4

5

6

7

8

9



python3

```
>>> for i in range(9):  
        print("i is: ", i)
```

i is: 0

i is: 1

i is: 2

i is: 3

i is: 4

i

0

1

2

3

4

5

6

7

8

9



python3

```
>>> for i in range(9):  
      print("i is: ", i)
```

```
i is: 0  
i is: 1  
i is: 2  
i is: 3  
i is: 4  
i is: 5
```

0

1

2

3

4

5

6

7

8

9

i



python3

```
>>> for i in range(9):  
        print("i is: ", i)
```

i is: 0

i is: 1

i is: 2

i is: 3

i is: 4

i is: 5

i is: 6

i

0

1

2

3

4

5

6

7

8

9



python3

```
>>> for i in range(9):  
      print("i is: ", i)
```

i is: 0

i is: 1

i is: 2

i is: 3

i is: 4

i is: 5

i is: 6

i is: 7

0

1

2

3

4

5

6

7

8

9

i



python3

```
>>> for i in range(9):  
        print("i is: ", i)
```

i is: 0

i is: 1

i is: 2

i is: 3

i is: 4

i is: 5

i is: 6

i is: 7

i is: 8

0

1

2

3

4

5

6

7

8

9

i



python3

```
>>> for i in range(9):  
        print("i is: ", i)
```

i is: 0

i is: 1

i is: 2

i is: 3

i is: 4

i is: 5

i is: 6

i is: 7

i is: 8

i is: 9

```
for i in range(2, 5):
```

2

3

4

0

1

2

3

4



python3

```
>>> for i in range(5):  
        print("i is: ", i)
```

0
1
2
3
4

0

1

2

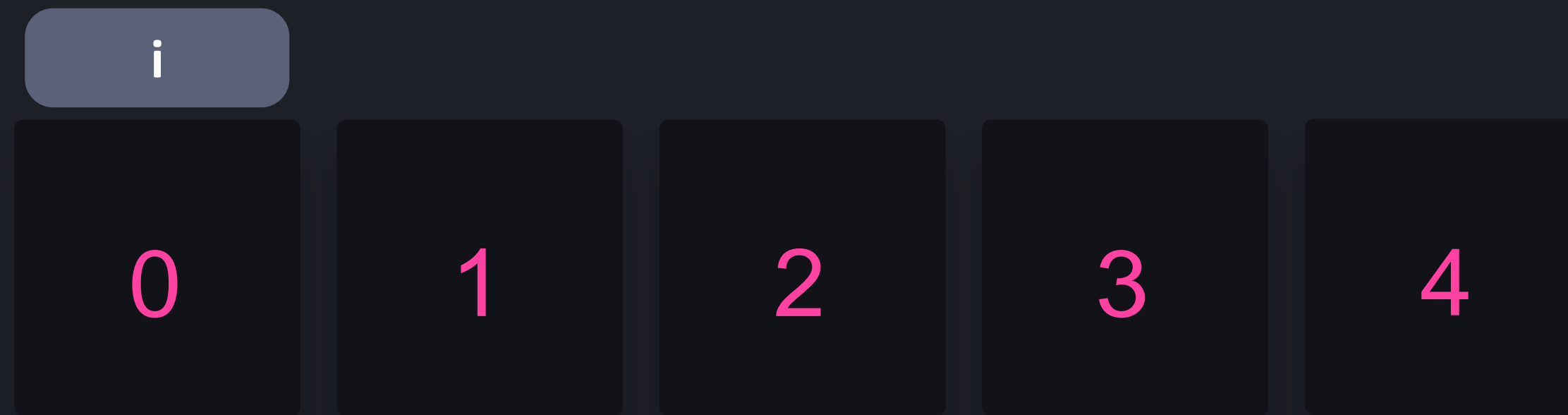
3

4



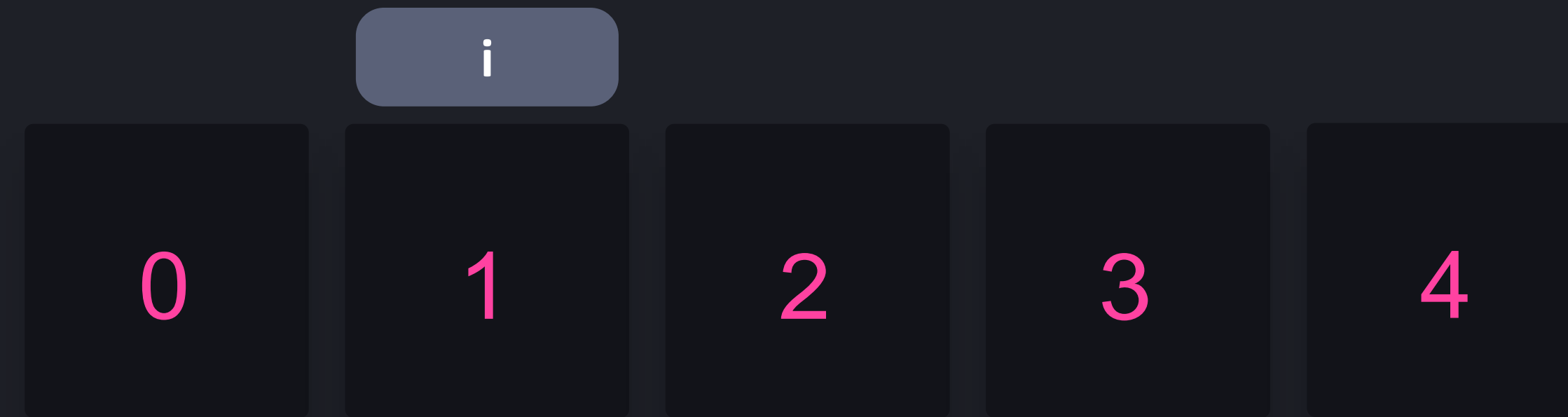
python3

```
>>> for i in range(5):  
    if(i == 2):  
        break  
    print("i is: ", i)
```



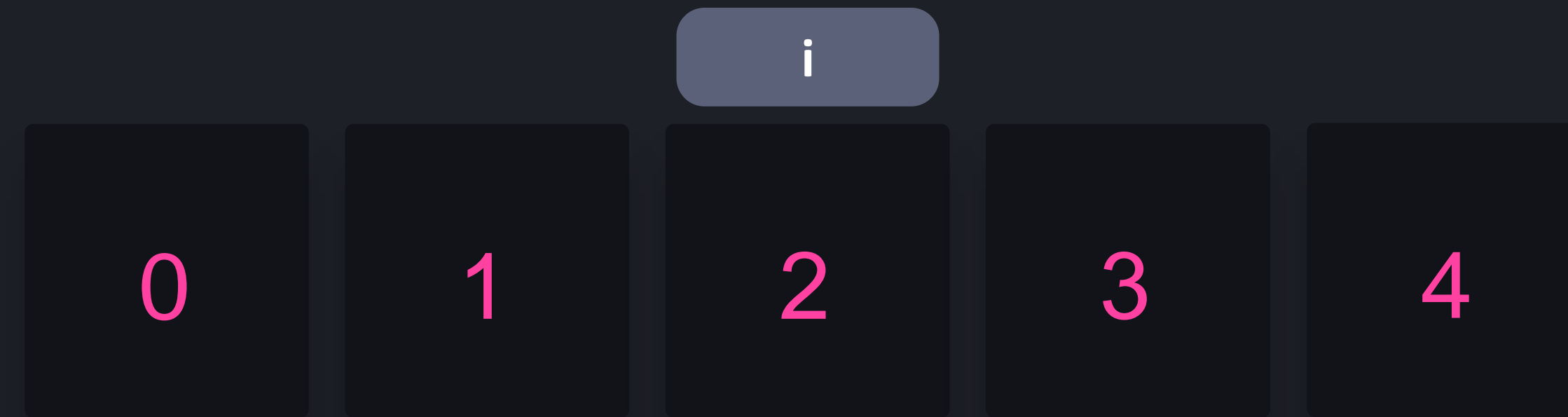
```
>>> for i in range(5):  
    if(i == 2):  
        break  
    print("i is: ", i)  
  
0
```

A terminal window titled "python3" with standard macOS window controls (red, yellow, green buttons). It displays a Python script that iterates over `range(5)`. The script prints the value of `i` for each iteration until it reaches 2, at which point it executes a `break` statement and terminates the loop. The output shown is the number 0, representing the first iteration.



```
>>> for i in range(5):  
    if(i == 2):  
        break  
    print("i is: ", i)  
  
0  
1
```

A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays a Python script that iterates over the range 0 to 4. The script prints the value of `i` for each iteration. The output shows the first two iterations, with the value 0 on the first line and 1 on the second line. The script is terminated by a blank line.



```
python3

>>> for i in range(5):
        if(i == 2):
            break
        print("i is: ", i)

0
1
>>>
```

0

1

2

3

4



python3

```
>>> for i in range(5):  
    print("i is: ", i)
```

0

1

2

3

4



python3

```
>>> for i in range(5):  
    if(i == 2):  
        continue  
    print("i is: ", i)
```

i

0

1

2

3

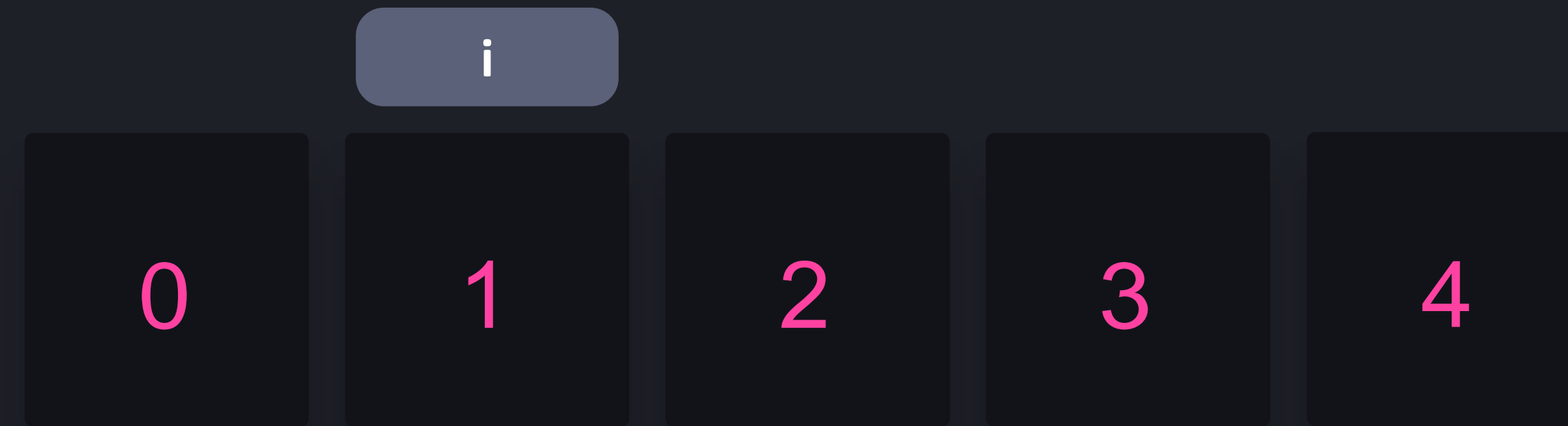
4



python3

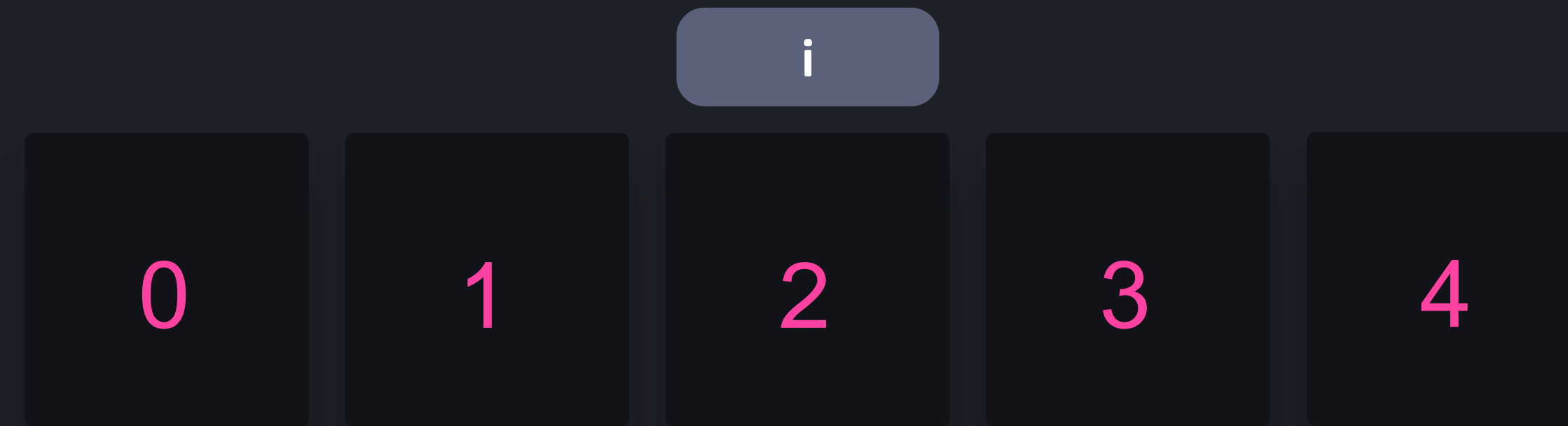
```
>>> for i in range(5):  
    if(i == 2):  
        continue  
    print("i is: ", i)
```

0



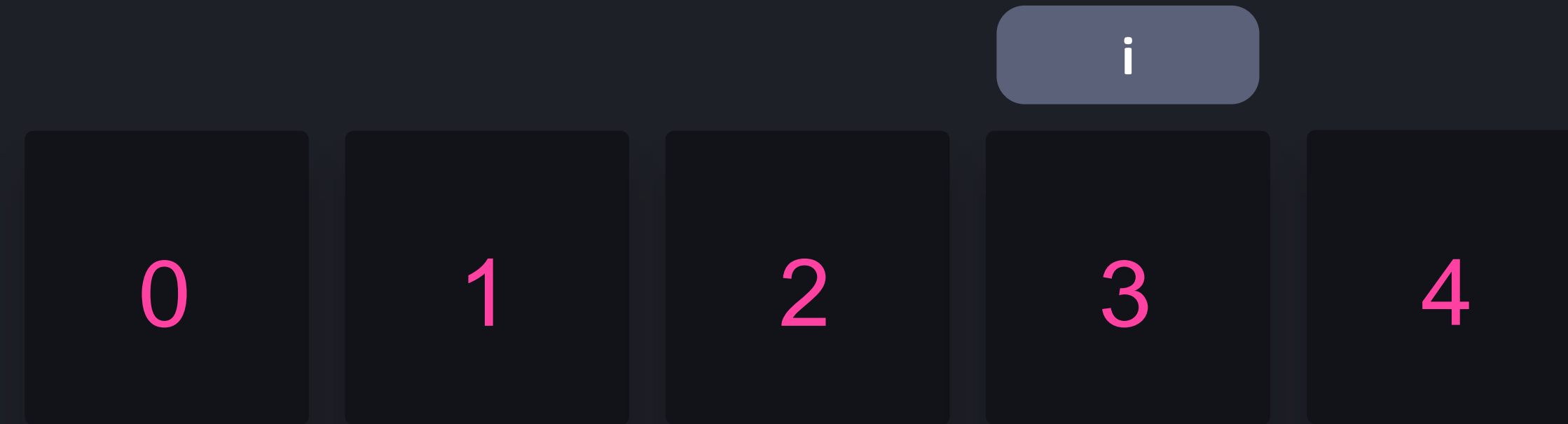
```
>>> for i in range(5):  
    if(i == 2):  
        continue  
    print("i is: ", i)  
  
0  
1
```

A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays a Python script that iterates over the range 0 to 4. For each value of `i`, it checks if `i` is equal to 2. If true, it executes `continue` and skips the rest of the loop body. Otherwise, it prints the value of `i`. The output shows the values 0 and 1, indicating that the iteration for `i=2` was skipped.



```
>>> for i in range(5):  
    if(i == 2):  
        continue  
    print("i is: ", i)  
  
0  
1
```

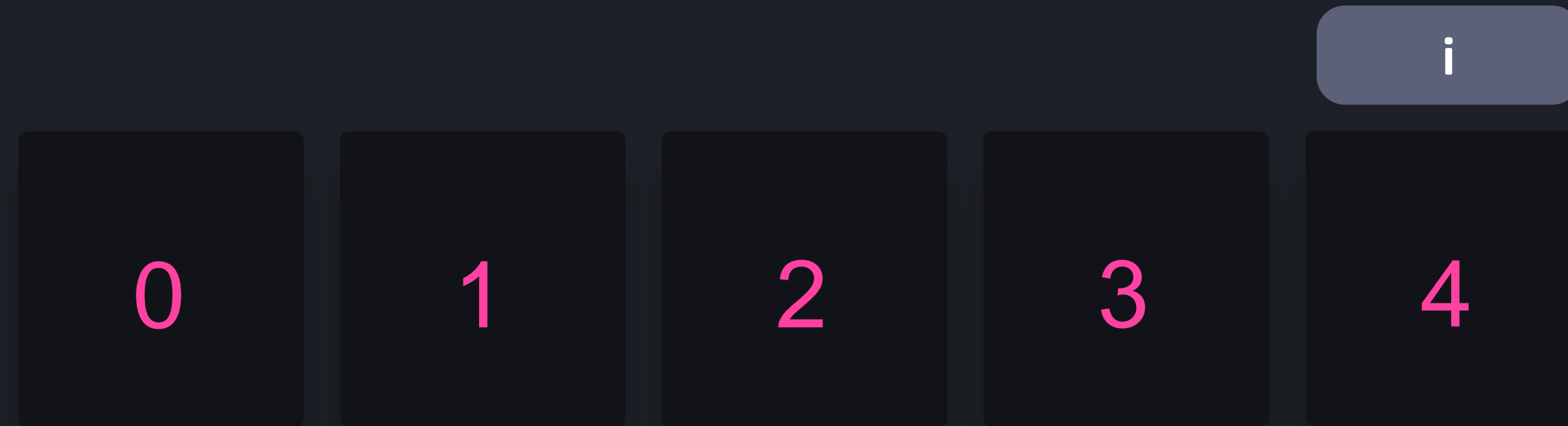
A terminal window titled "python3" with standard macOS window controls (red, yellow, green buttons). It displays a Python script that iterates over `range(5)`. For each value of `i`, it checks if `i` is equal to 2. If true, it executes `continue` and skips the `print` statement. If false, it prints the value of `i`. The output shows the numbers 0 and 1, indicating that the loop has started and the first iteration is complete.



A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays a Python script that iterates over the range 0 to 4. When the index 'i' is 2, the 'continue' statement is executed, skipping the 'print' statement for that iteration. The output shows the values 0, 1, and 3, with the value 2 being skipped.

```
>>> for i in range(5):  
    if(i == 2):  
        continue  
    print("i is: ", i)
```

0
1
3



```
python3

>>> for i in range(5):
        if(i == 2):
            continue
        print("i is: ", i)

0
1
3
4
```

- `if/else` statements allow us to conditionally run code
- A `while` loop makes it possible to repetitively execute code based on a certain condition
- We can execute code for each item in a sequence with a `for ... in` loop



KodeKloud



Operators



python3

```
>>> age1 = 24
```



python3

```
>>> age1 = 24
```

```
>>> age2 = 16
```




python3

```
>>> age1 = 24
>>> age2 = 16
>>> if( Both ages are higher than 18 ):
    print("You are both adults")
elif( One age is higher than 18 ):
    print("One of you is an adult")
else:
    print("You are both children")
```



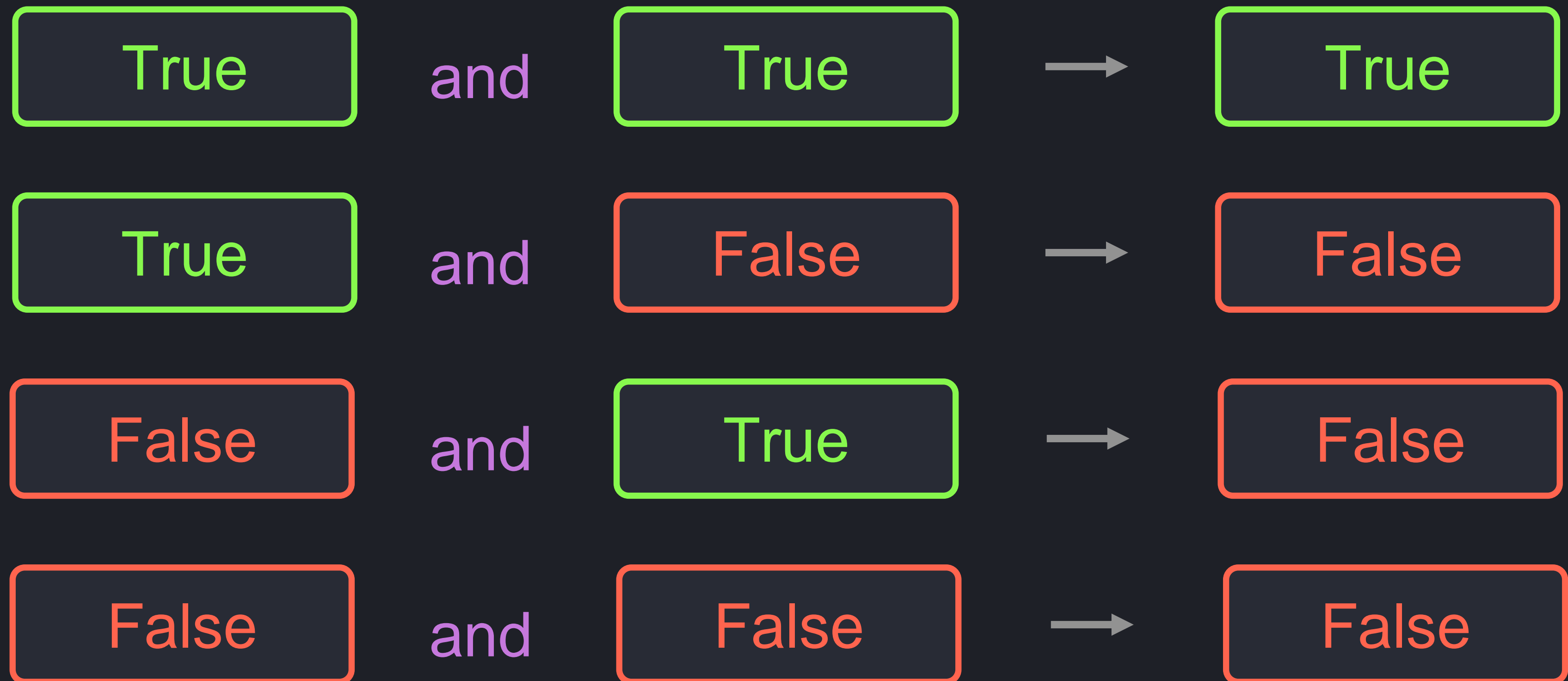
python3

```
>>> age1 = 24
>>> age2 = 16
>>> if( Both ages are higher than 18 ):
    print("You are both adults")
elif( One age is higher than 18 ):
    print("One of you is an adult")
else:
    print("You are both children")
```



python3

```
>>> age1 = 24
>>> age2 = 16
>>> if( age1 >= 18 and age2 >= 18 ):
    print("You are both adults")
elif( One age is higher than 18 ):
    print("One of you is an adult")
else:
    print("You are both children")
```





python3

```
>>> age1 = 24
>>> age2 = 16
>>> if( age1 >= 18 and age2 >= 18 ):
    print("You are both adults")
elif( One age is higher than 18 ):
    print("One of you is an adult")
else:
    print("You are both children")
```



python3

```
>>> age1 = 24
>>> age2 = 16
>>> if( age1 >= 18 and age2 >= 18 ):
    print("You are both adults")
elif( One age is higher than 18 ):
    print("One of you is an adult")
else:
    print("You are both children")
```



python3

```
>>> age1 = 24
>>> age2 = 16
>>> if( age1 >= 18 and age2 >= 18 ):
    print("You are both adults")
elif( age1 >= 18 or age2 >= 18 ):
    print("One of you is an adult")
else:
    print("You are both children")
```



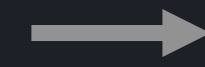


python3

```
>>> age1 = 24
>>> age2 = 16
>>> if( age1 >= 18 and age2 >= 18 ):
    print("You are both adults")
elif( age1 >= 18 or age2 >= 18 ):
    print("One of you is an adult")
else:
    print("You are both children")
```

not

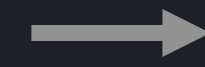
True



False

not

False



True



python3

```
>>> is_hungry = False
>>> if( not is_hungry ):
    print("You are not hungry")
```



python3

```
>>> is_hungry = False
>>> if( not is_hungry ):
    print("You are not hungry")
"You are not hungry"
```



KodeKloud



Bitwise Operators

Logical Operators

or

and

not

Bitwise Operators



&

Conjunction



|

Disjunction



~

Negation



^

Exclusive

Bitwise Operators

&

Two 1



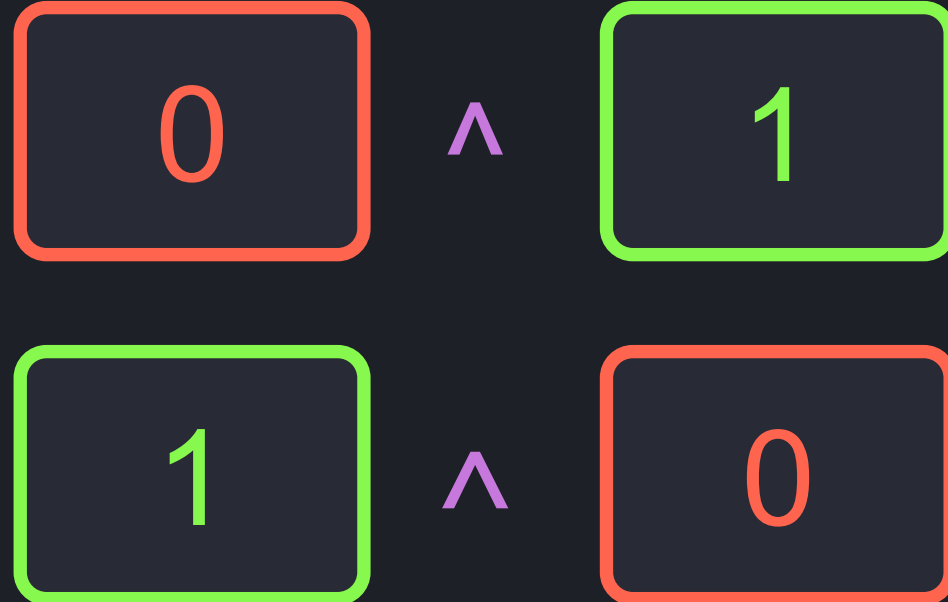
|

At least one 1



^

Exactly 1





python3

```
>>> 15 & 22
```



python3

```
>>> print(bin(15))
```

```
0b1111
```



python3

```
>>> print(bin(15))
```

```
0b1111
```

```
>>> print(bin(22))
```

```
0b10110
```



python3

```
>>> print(bin(15))
```

```
0b1111
```

```
>>> print(bin(22))
```

```
0b10110
```

15

0

0

0

0

1

1

1

1

22

0

0

0

1

0

1

1

0

15

22

0

0

0

0

1

1

1

1

0

0

0

1

0

1

1

0

15

22

0

0

0

0

1

1

1

1

0

0

0

1

0

1

1

0

&

Two 1

1

&

1

15

22

0

0

0

0

1

1

1

1



0

0

0

1

0

1

1

0

0

0

0

0

0

1

1

0

15

22

0

0

0

0

1

1

1

1



0

0

0

1

0

1

1

0

0

0

0

0

0

1

1

0

15

22

6

0

0

0

0

1

1

1

1



0

0

0

1

0

1

1

0

0

0

0

0

0

1

1

0



python3

```
>>> print(15 & 22)
```



python3

```
>>> print(15 & 22)
```

```
6
```



python3

```
>>> print(15 | 22)
```

15

22

0

0

0

0

1

1

1

1

0

0

0

1

0

1

1

0

15

0

0

0

|

0

1

1

1

1

22

0

0

0

At least one 1

1

0

1

1

0

1

|

1

0

|

1

1

|

0

15

22

0

0

0

0

1

1

1

1



0

0

0

1

0

1

1

0

0

0

0

1

1

1

1

1

15

22

31

0

0

0

0

1

1

1

1



0

0

0

1

0

1

1

0

0

0

0

1

1

1

1

1



python3

```
>>> print(15 | 22)
```



python3

```
>>> print(15 | 22)
```

```
31
```



python3

```
>>> print(15 ^ 22)
```

15

22

0

0

0

0

1

1

1

1

0

0

0

1

0

1

1

0

15

22

0

0

0

0

1

1

1

1

\wedge

0

0

0

1

0

1

1

0

Exactly 1

0

\wedge

1

1

\wedge

0

15

22

0

0

0

0

1

1

1

1



0

0

0

1

0

1

1

0

0

0

0

1

1

1

0

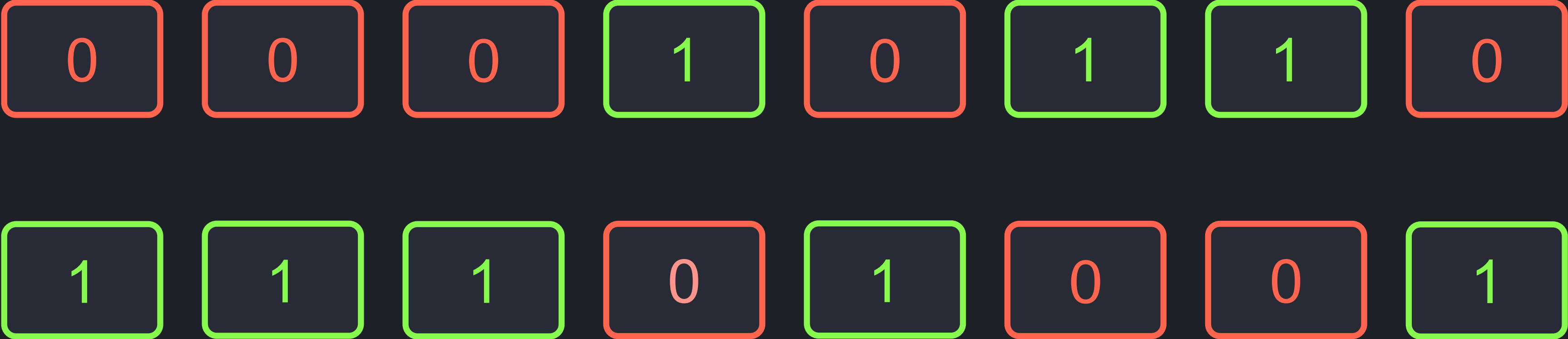
1



python3

```
>>> print(~22)
```

22



22

-23

0

0

0

1

0

1

1

0

1

1

1

0

1

0

0

1

Without Shortcut Operator

```
bit1 = bit1 & 22
```

```
bit1 = bit1 | 22
```

```
bit1 = bit1 ^ 22
```

With Shortcut Operator

```
bit1 &= 22
```

```
bit1 |= 22
```

```
bit1 ^= 22
```

Bit Shifting

A dark blue square button with rounded corners containing the text ">>" in a light blue monospace font, representing a right bit shift operation.

>>

Bit Shift Right

A dark blue square button with rounded corners containing the text "<<" in a light blue monospace font, representing a left bit shift operation.

<<

Bit Shift Left



python3

```
>>> print(22 >> 1)
```



python3

```
>>> print(22 >> 1)
```

22

0

0

0

1

0

1

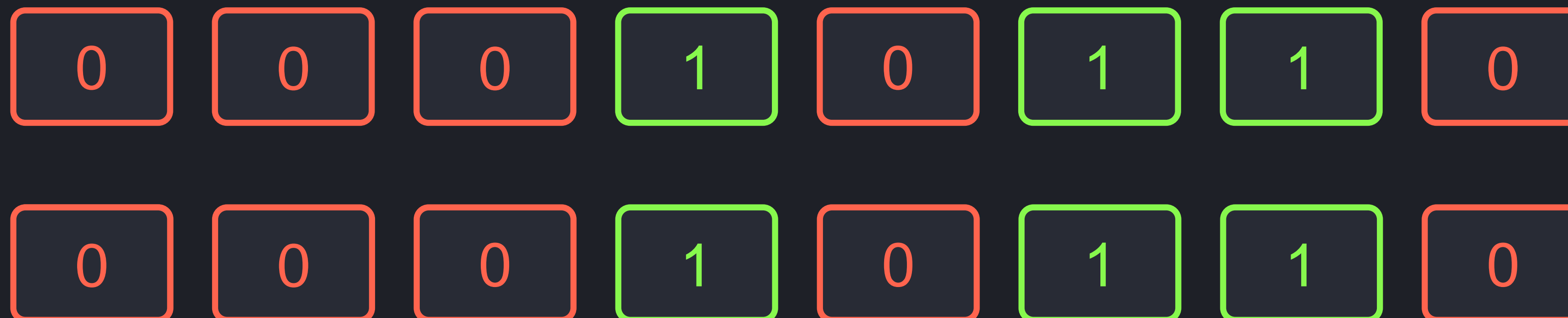
1

0

python3

```
>>> print(22 >> 1)
```

22





python3

```
>>> print(22 >> 1)
```

22

0

0

0

1

0

1

1

0

0

0

0

0

1

0

1

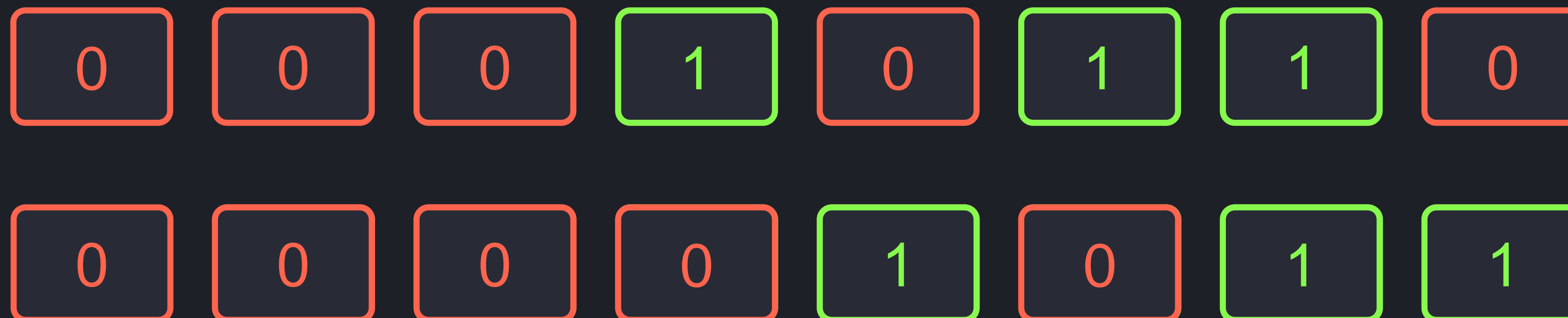
1

python3

```
>>> print(22 >> 1)
```

22

11

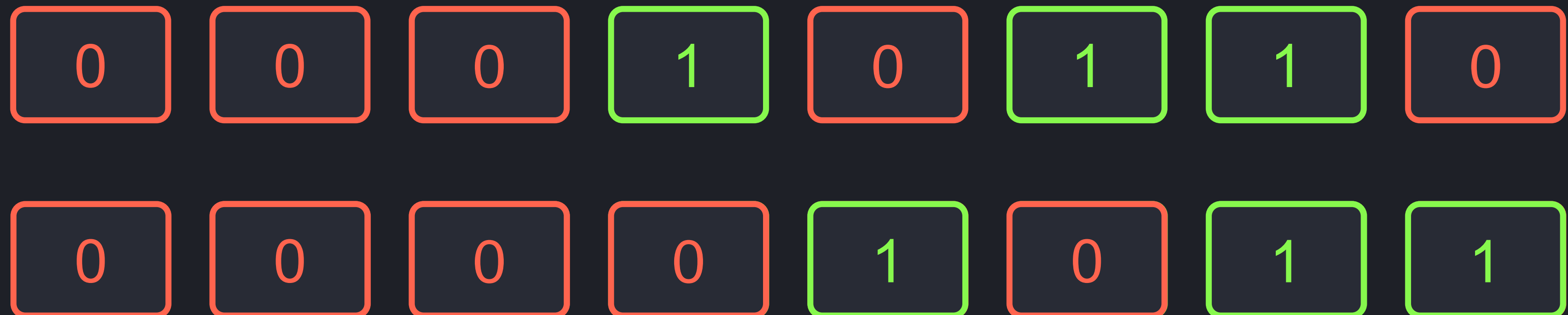


python3

```
>>> print(22 >> 1)
11
```

22

11





python3

```
>>> print(22 >> 2)
```

22

0

0

0

1

0

1

1

0

0

0

0

1

0

1

1

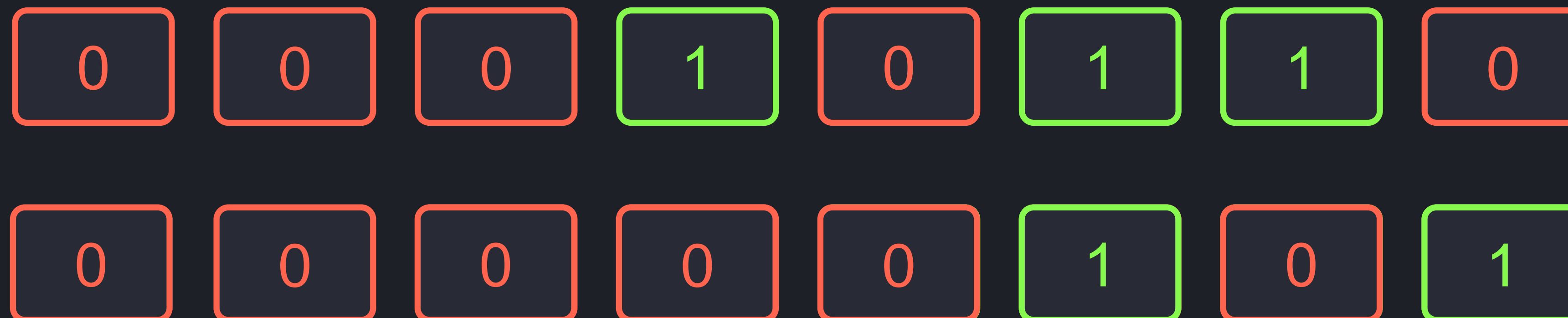
0

python3

```
>>> print(22 >> 2)
5
```

22

5





python3

```
>>> print(22 << 1)
```

22

0

0

0

1

0

1

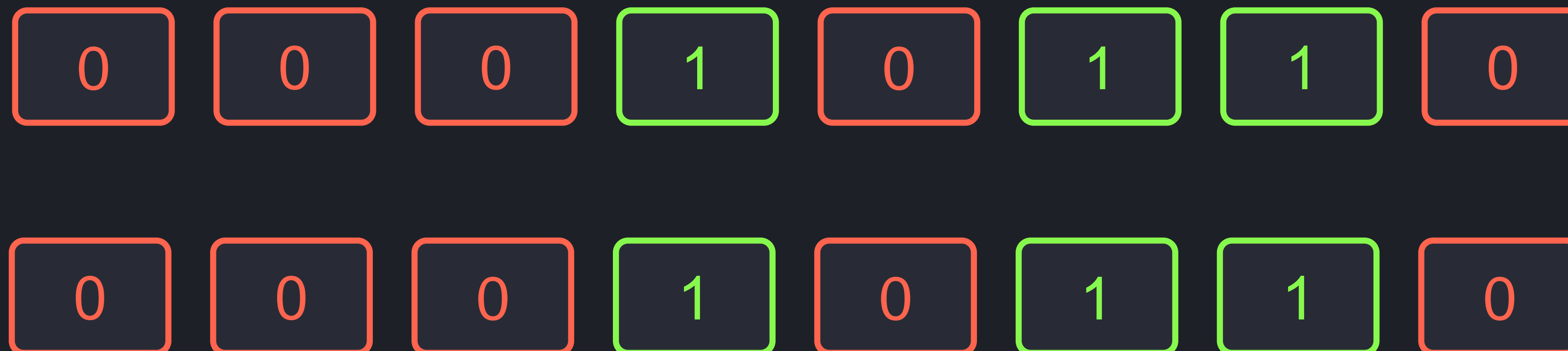
1

0

python3

```
>>> print(22 << 1)
```

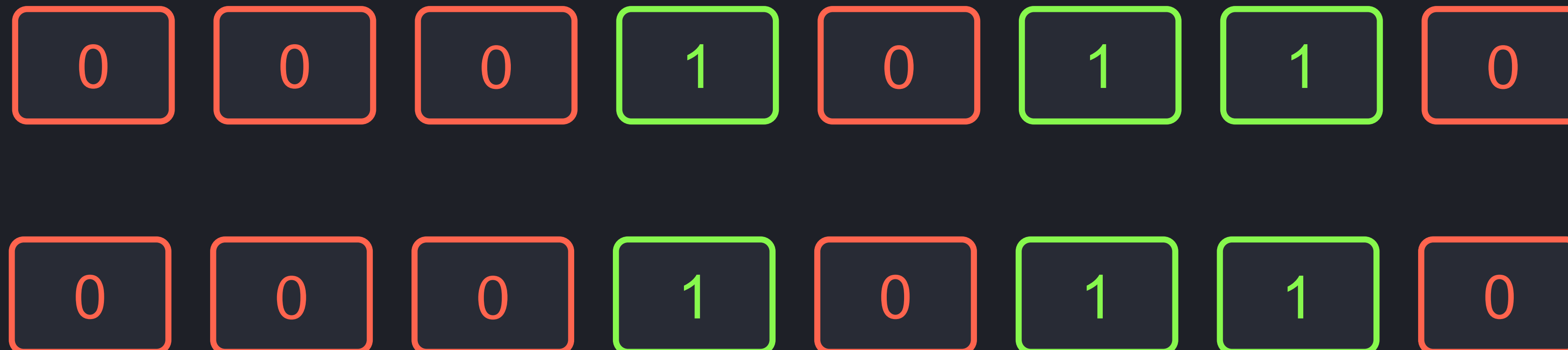
22



python3

```
>>> print(22 << 1)
```

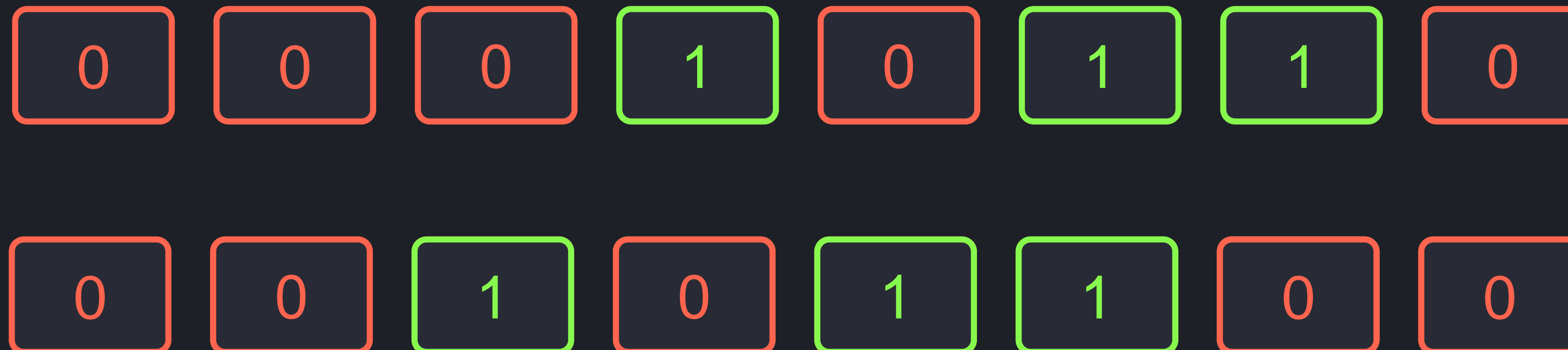
22



python3

```
>>> print(22 << 1)
```

22



python3

```
>>> print(22 << 1)
```

22



44



python3

```
>>> print(22 << 1)
```

22



44



```
print(22 // 2)
```



```
print(22 >> 1)
```

```
print(22 // 4)
```



```
print(22 >> 2)
```

```
print(22 * 2)
```



```
print(22 << 1)
```

```
print(22 * 4)
```



```
print(22 << 2)
```

Operators

- Logical operators **and** **not** and **or** return boolean values based on the passed values
- Bitwise operators **&** **|** **^** and **~** allow us to manipulate single bits of data, and return **0** or **1** based on the value of the bits that are used
- Bit shifting can be done with the **<<** and **>>** operators



KodeKloud

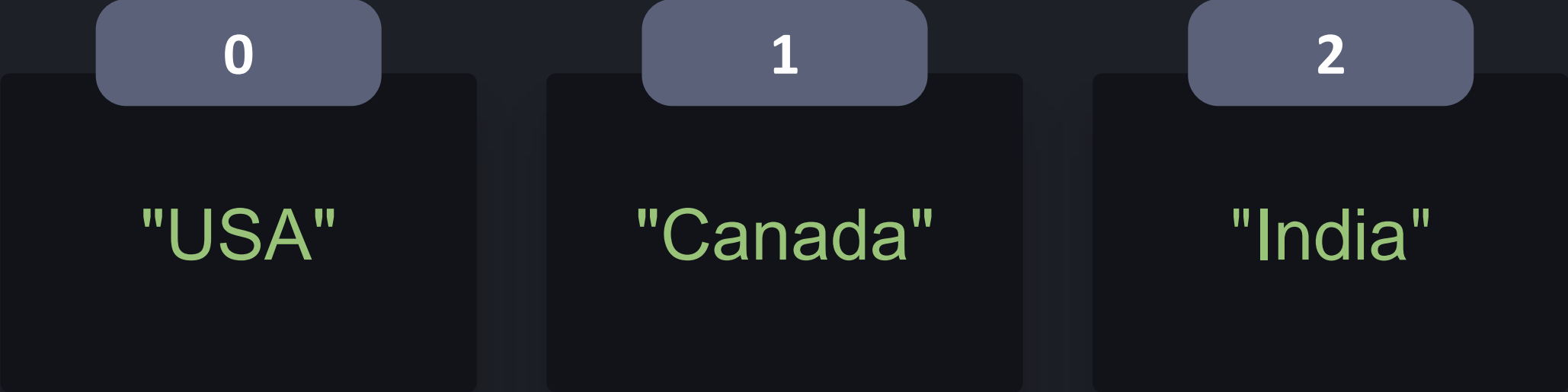


Lists



python3

```
>>> countries = ["USA", "Canada", "India"]
```



python3

```
>>> countries = ["USA", "Canada", "India"]
```

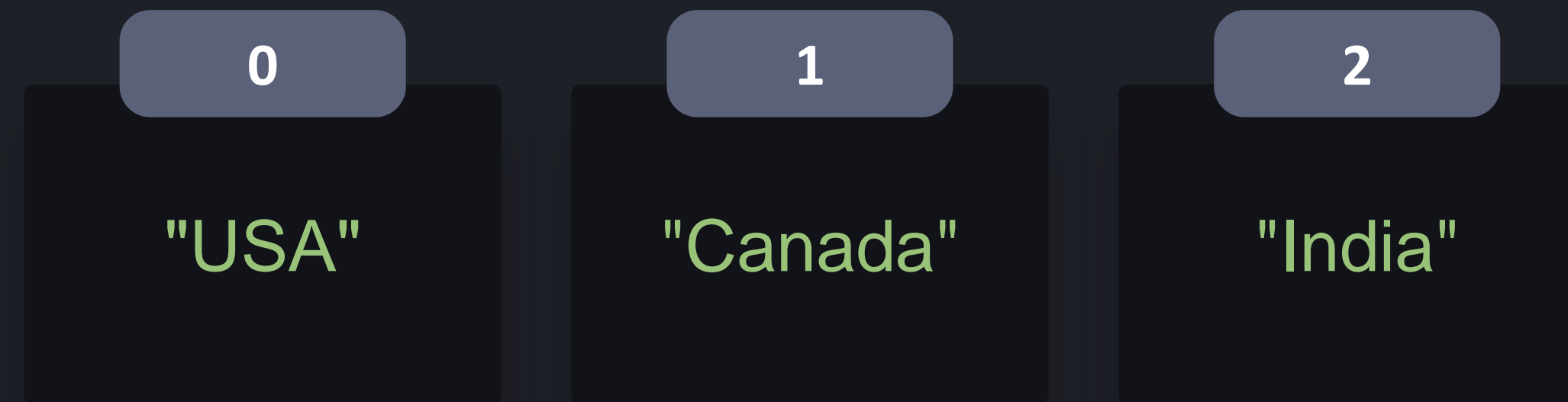
```
>>> print(countries[0])
```

```
>>> print(countries[1])
```

Canada

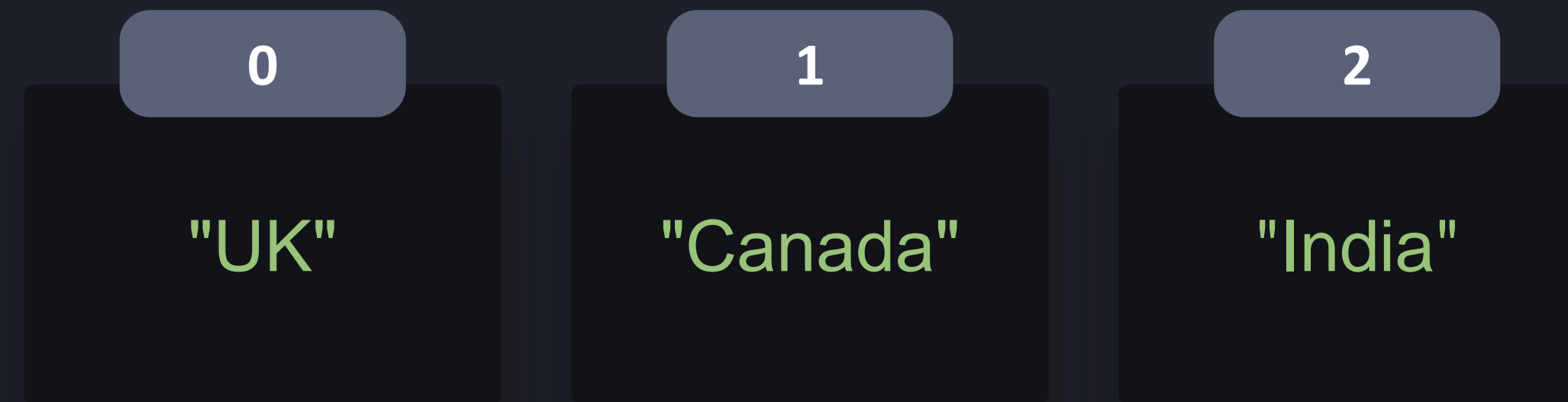
```
>>> print(countries[2])
```

India



A terminal window titled 'python3' with three colored window control buttons (red, yellow, green) on the left. The terminal displays two lines of Python code:

```
>>> countries = ["USA", "Canada", "India"]
>>> countries[0] = "UK"
```



A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays two lines of Python code. The first line creates a list named "countries" with elements "USA", "Canada", and "India". The second line updates the first element of the list (at index 0) to "UK".

```
>>> countries = ["USA", "Canada", "India"]
>>> countries[0] = "UK"
```

len()



python3

```
>>> countries = ["USA", "Canada", "India"]  
>>> len(countries)
```

len()

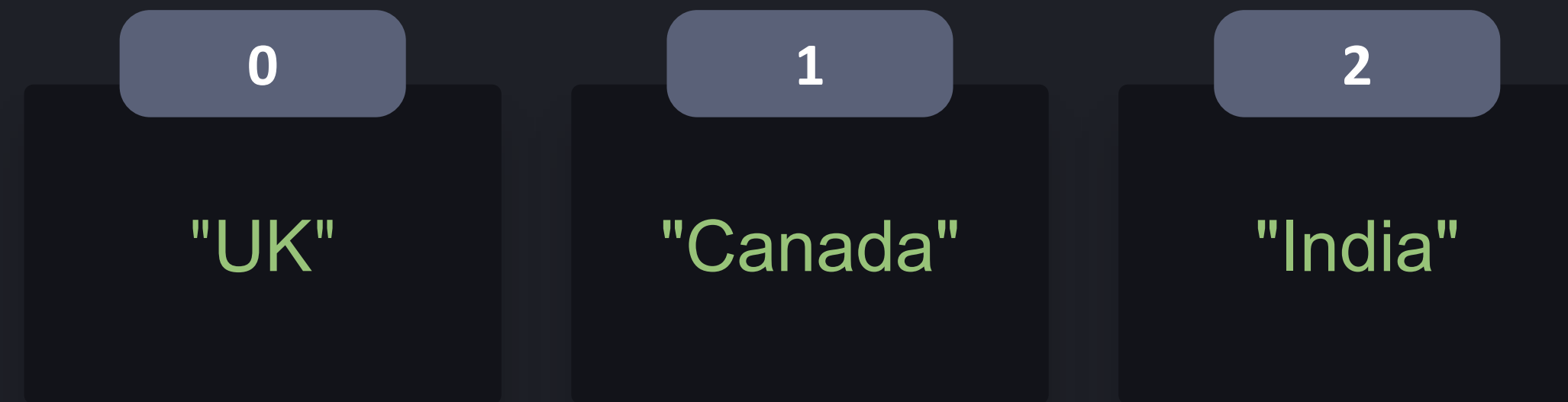


python3

```
>>> countries = ["USA", "Canada", "India"]
```

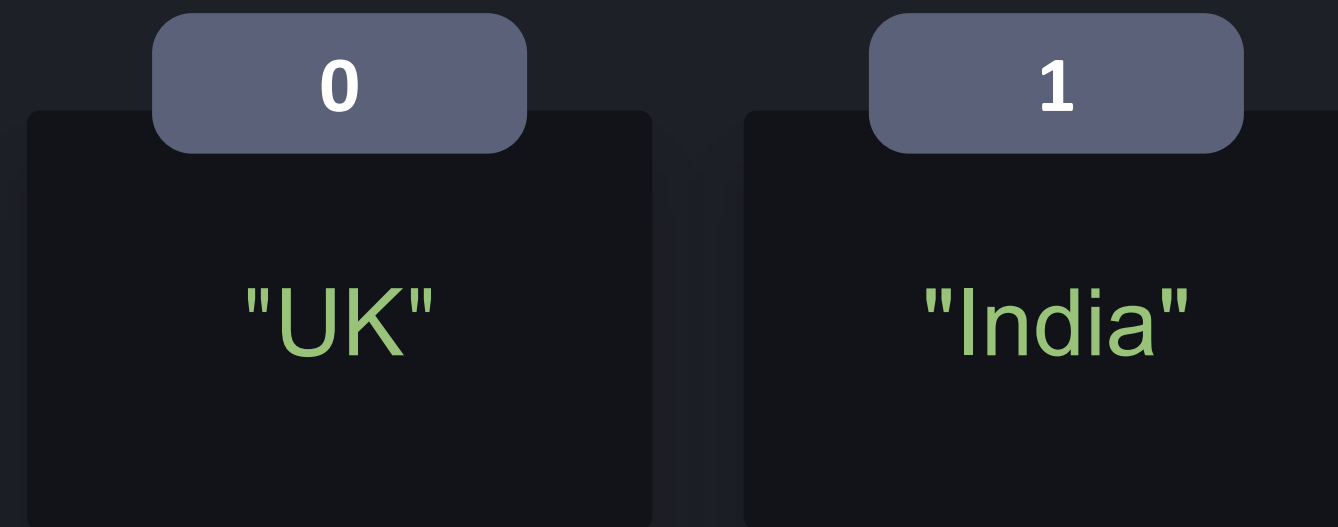
```
>>> len(countries)
```

```
3
```

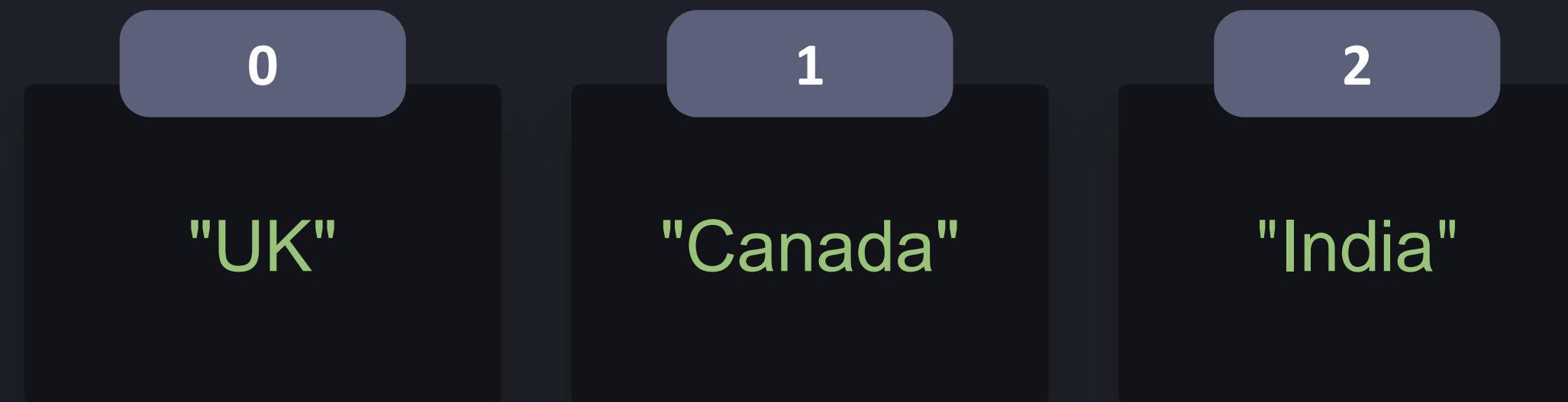


A terminal window titled 'python3' with three colored window control buttons (red, yellow, green) on the left. The terminal displays two lines of Python code:

```
>>> countries = ["USA", "Canada", "India"]
>>> del countries[1]
```

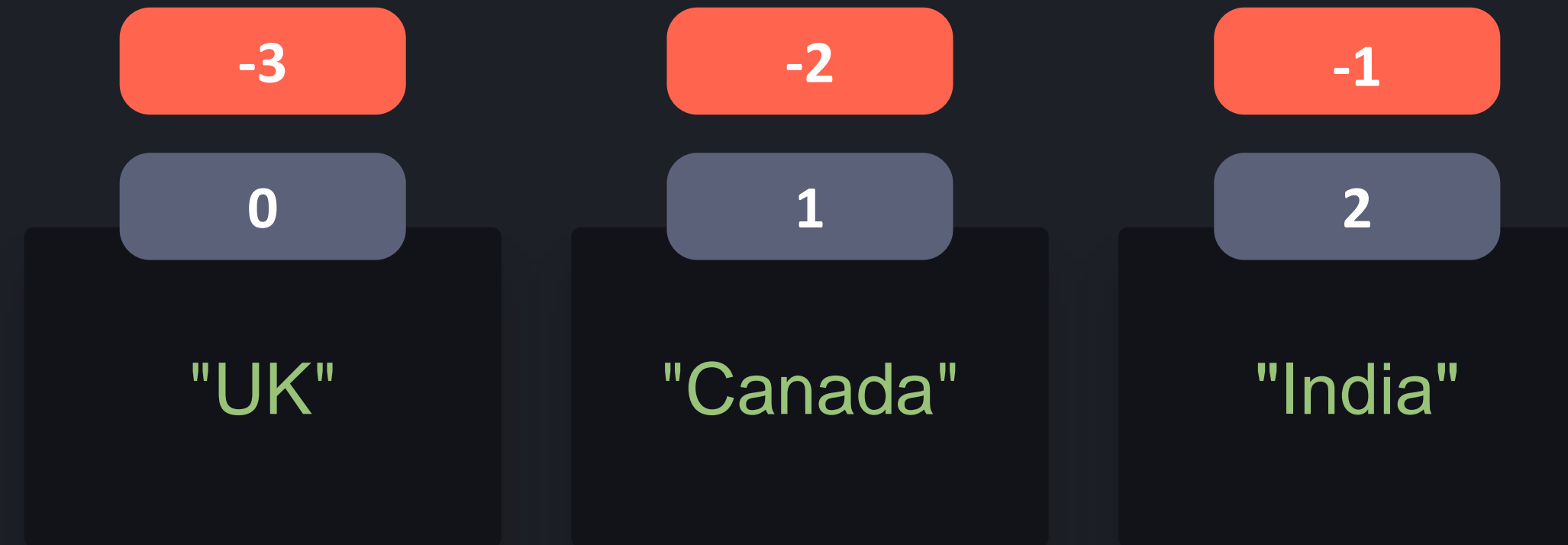


A screenshot of a Python 3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text 'python3' in the center. The main area of the window is dark gray and contains two lines of code in a light green font. The first line is `>>> countries = ["USA", "Canada", "India"]` and the second line is `>>> del countries[1]`.



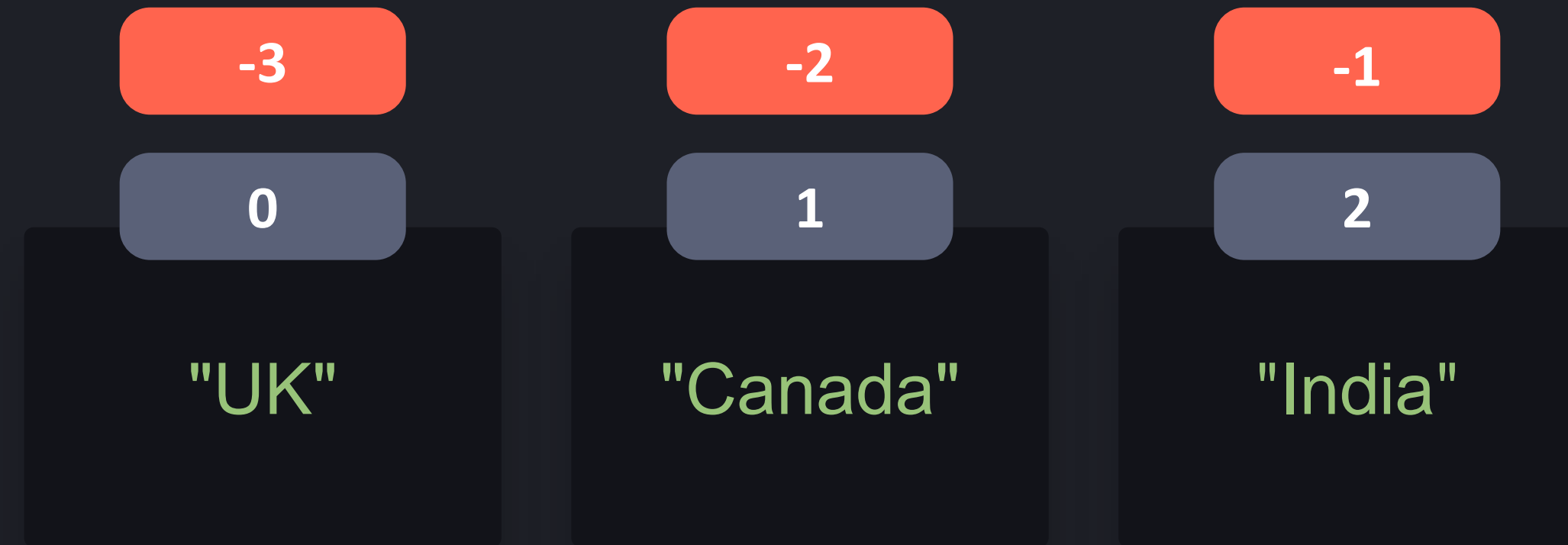
A screenshot of a Python 3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text 'python3' in the center. The main area of the window is dark gray and contains a single line of Python code: `>>> countries = ["USA", "Canada", "India"]`. The text is in a light green monospace font.

```
>>> countries = ["USA", "Canada", "India"]
```



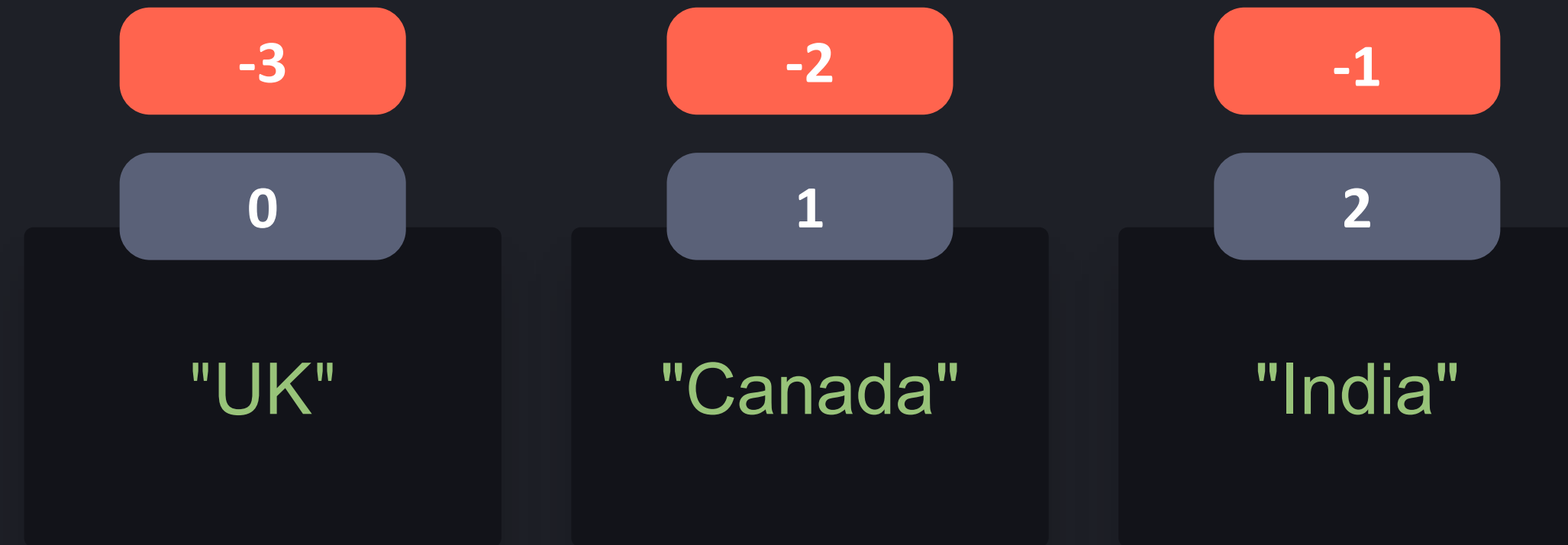
A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays a Python list assignment:

```
>>> countries = ["USA", "Canada", "India"]
```



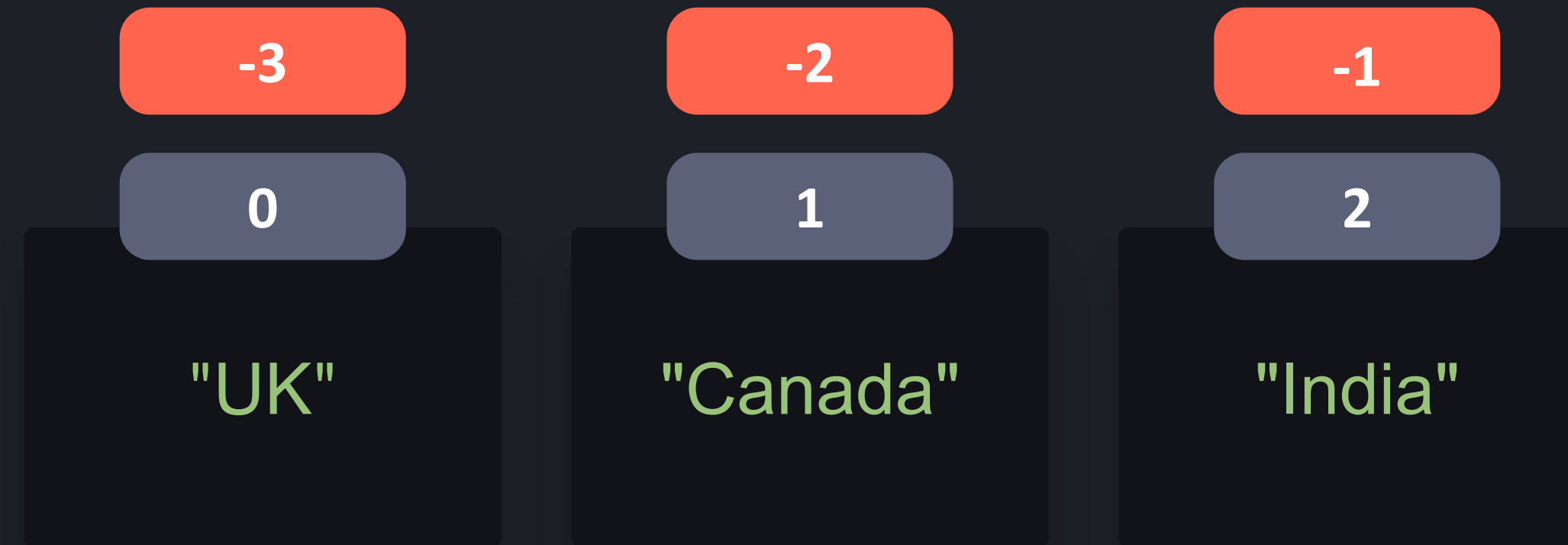
```
python3

>>> countries = ["USA", "Canada", "India"]
>>> print(countries[-1])
```



```
python3

>>> countries = ["USA", "Canada", "India"]
>>> print(countries[-1])
"India"
```



python3

```
>>> countries = ["USA", "Canada", "India"]  
>>> print(countries[4])
```

IndexError: list index out of range



KodeKloud



Lists - Methods

```
list.append()
```

```
list.insert()
```


Functions

`print()`

`len()`

`input()`

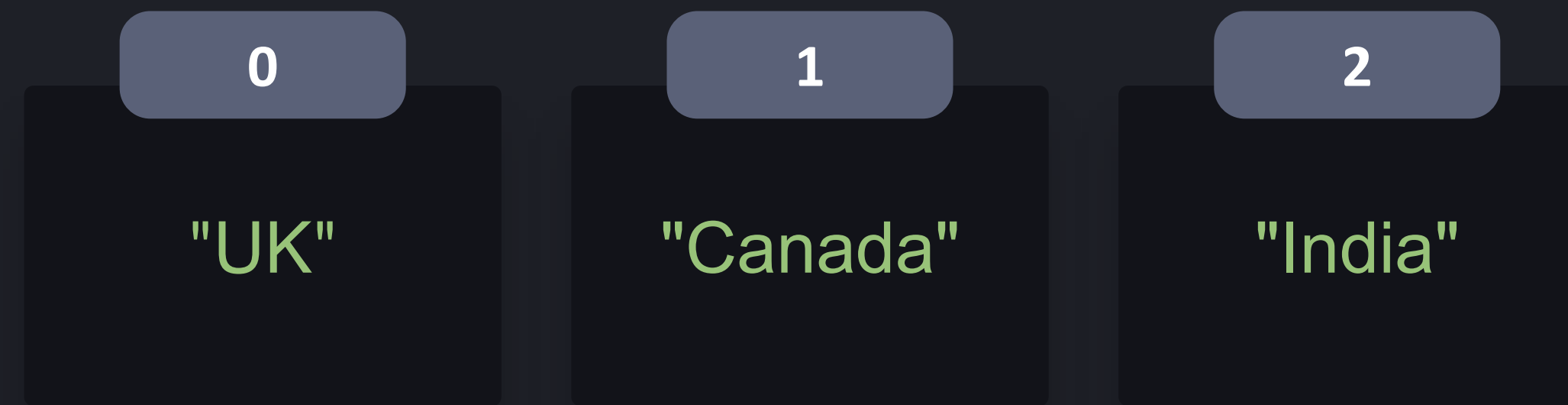
Methods

`list.append()`

`list.insert()`

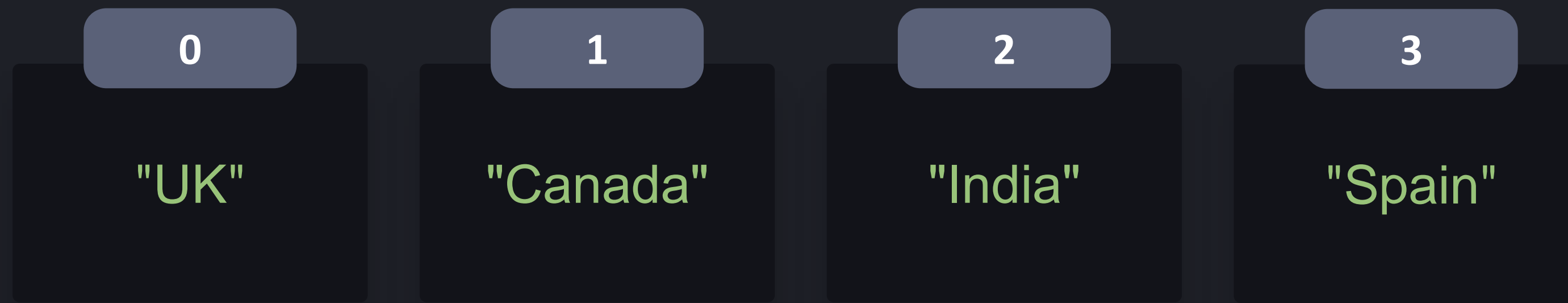
```
list.append()
```

```
list.insert()
```



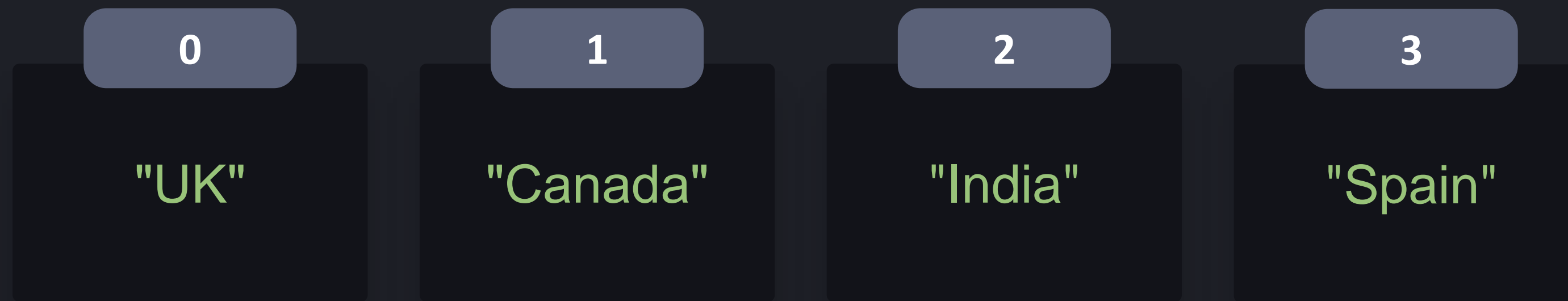
A screenshot of a Python 3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text 'python3' in the center. The main area of the window is dark gray and contains two lines of Python code in a light green monospace font. The first line is `>>> countries = ["USA", "Canada", "India"]` and the second line is `>>> countries.append("Spain")`.

```
>>> countries = ["USA", "Canada", "India"]
>>> countries.append("Spain")
```



A terminal window titled 'python3' with a dark gray background. The window has a title bar with three colored circles (red, yellow, green) on the left. The terminal displays two lines of Python code in a light green monospace font:

```
>>> countries = ["USA", "Canada", "India"]
>>> countries.append("Spain")
```



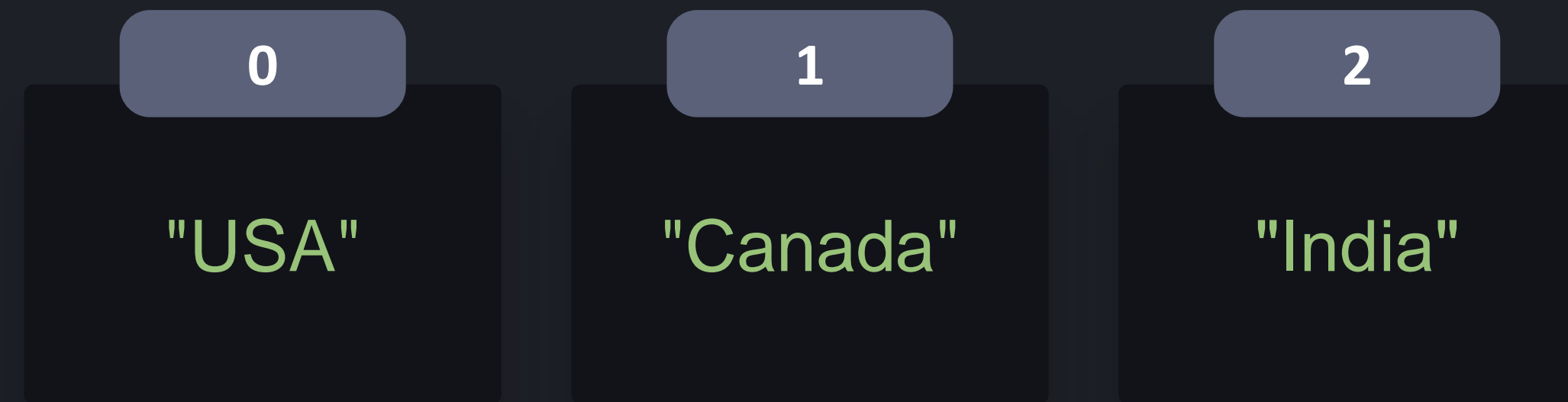
A terminal window titled 'python3' with three colored window control buttons (red, yellow, green) on the left. The terminal displays three lines of Python code in a light green monospace font:

```
>>> countries = ["USA", "Canada", "India"]
>>> countries.append("Spain")
>>> countries.insert(2, "Italy")
```



A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays three lines of Python code that create a list, append an element, and insert an element at a specific index.

```
>>> countries = ["USA", "Canada", "India"]
>>> countries.append("Spain")
>>> countries.insert(2, "Italy")
```



A screenshot of a Python3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text 'python3' in the center. The main area of the window is dark gray and contains a single line of Python code: `>>> countries = ["USA", "Canada", "India"]`. The code is displayed in a monospaced font with syntax highlighting: the prompt `>>>` is white, the variable `countries` is orange, the equals sign is white, and the list elements are in quotes and green.

```
>>> countries = ["USA", "Canada", "India"]
```

temp

"USA"

0

"USA"

1

"Canada"

2

"India"

python3

```
>>> countries = ["USA", "Canada", "India"]  
>>> temp = countries[0]
```


temp

"USA"

0

"USA"

1

"Canada"

2

"India"

python3

```
>>> countries = ["USA", "Canada", "India"]
>>> temp = countries[0]
>>> countries[0] = countries[1]
```

temp

"USA"

0

"Canada"

1

"Canada"

2

"India"



python3

```
>>> countries = ["USA", "Canada", "India"]  
>>> temp = countries[0]  
>>> countries[0] = countries[1]
```

temp

"USA"

0

"Canada"

1

"Canada"

2

"India"



python3

```
>>> countries = ["USA", "Canada", "India"]
>>> temp = countries[0]
>>> countries[0] = countries[1]
>>> countries[1] = temp
```

temp

"USA"

0

"Canada"

1

"USA"

2

"India"



python3

```
>>> countries = ["USA", "Canada", "India"]
>>> temp = countries[0]
>>> countries[0] = countries[1]
>>> countries[1] = temp
```

temp

"USA"

0

"Canada"

1

"USA"

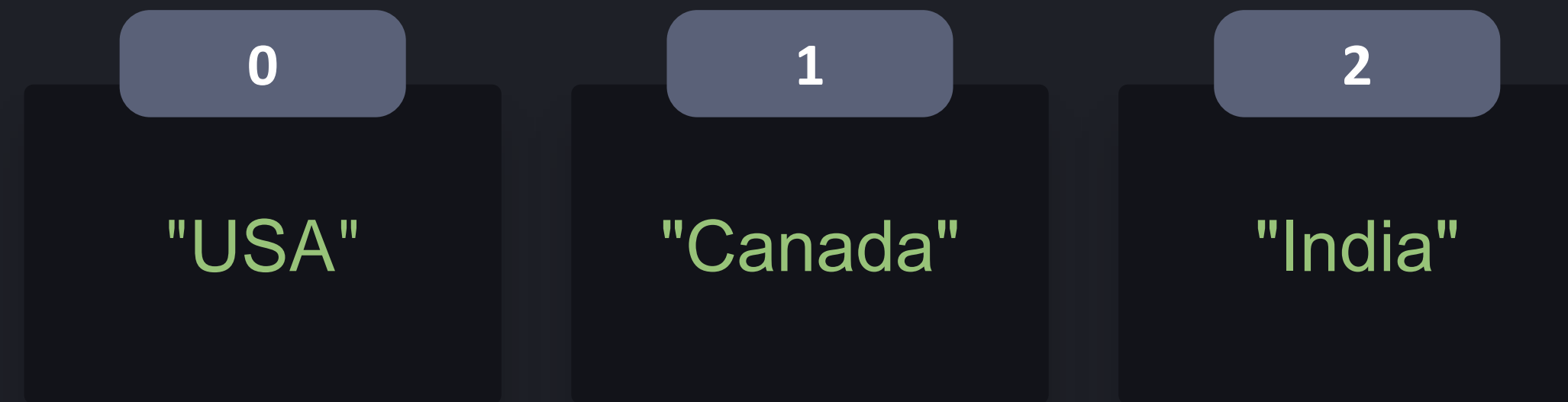
2

"India"



python3

```
>>> countries = ["USA", "Canada", "India"]
>>> temp = countries[0]
>>> countries[0] = countries[1]
>>> countries[1] = temp
```



python3

```
>>> countries = ["USA", "Canada", "India"]  
>>> countries[0], countries[1] = countries[1], countries[0]
```

0

"USA"

1

"Canada"

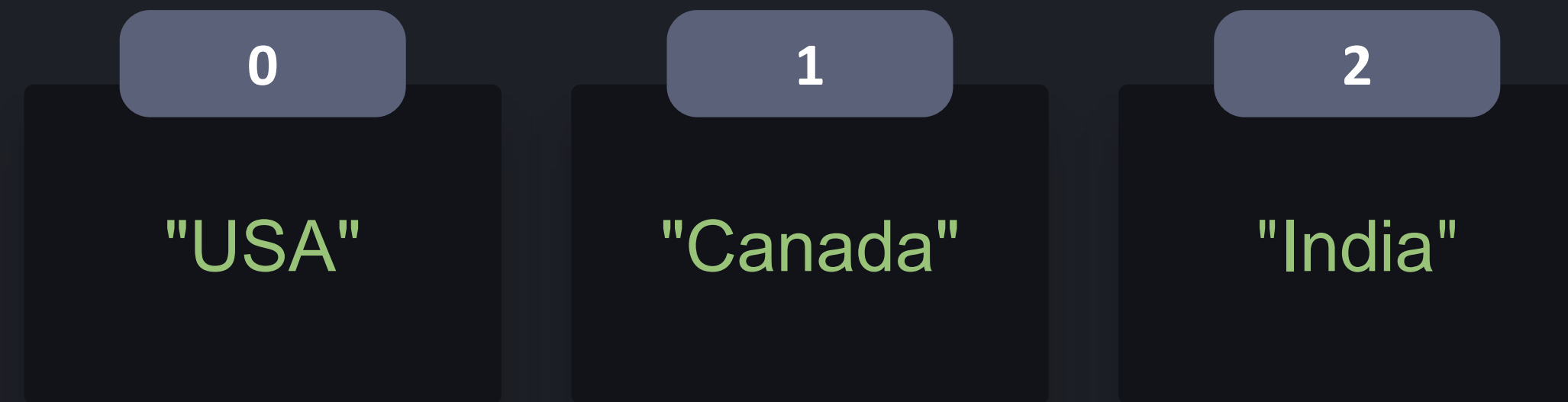
2

"India"

python3
countries[0], countries[1] = countries[1], countries[0]

>>> countries = ["USA", "Canada", "India"]

>>>



python3

```
>>> countries = ["USA", "Canada", "India"]  
>>> countries[0], countries[1] = countries[1], countries[0]
```


0

"Canada"

1

"USA"

2

"India"



python3

```
>>> countries = ["USA", "Canada", "India"]
```

```
>>> countries[0], countries[1] = countries[1], countries[0]
```

```
list.sort()
```

```
list.reverse()
```



```
python3

>>> ages = [56, 72, 24, 46]
>>> ages.sort()
```



A terminal window titled 'python3' with three colored window control buttons (red, yellow, green) on the left. The terminal displays the following Python code:

```
>>> ages = [56, 72, 24, 46]
>>> ages.sort()
>>> print(ages)
```



A screenshot of a Python terminal window titled 'python3'. The window has a dark gray background and a title bar with three colored circles (red, yellow, green) on the left. The terminal shows the following code and output:

```
>>> ages = [56, 72, 24, 46]
>>> ages.sort()
>>> print(ages)
[24, 46, 56, 72]
```



```
python3

>>> ages = [56, 72, 24, 46]
>>> ages.reverse()
```



A screenshot of a Python 3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text 'python3' in the center. The main area of the window is dark gray and contains three lines of Python code. The first line is '>>> ages = [56, 72, 24, 46]', the second is '>>> ages.reverse()', and the third is '>>> print(ages)'. The code is color-coded: '>>>' is white, 'ages' is yellow, '[' is white, the numbers are pink/magenta, ',' is white, ']' is white, '.' is white, 'reverse()' is purple, 'print' is cyan, and 'ages' is yellow.

```
>>> ages = [56, 72, 24, 46]
>>> ages.reverse()
>>> print(ages)
```



```
python3

>>> ages = [56, 72, 24, 46]
>>> ages.reverse()
>>> print(ages)
[46, 24, 72, 56]
```




KodeKloud

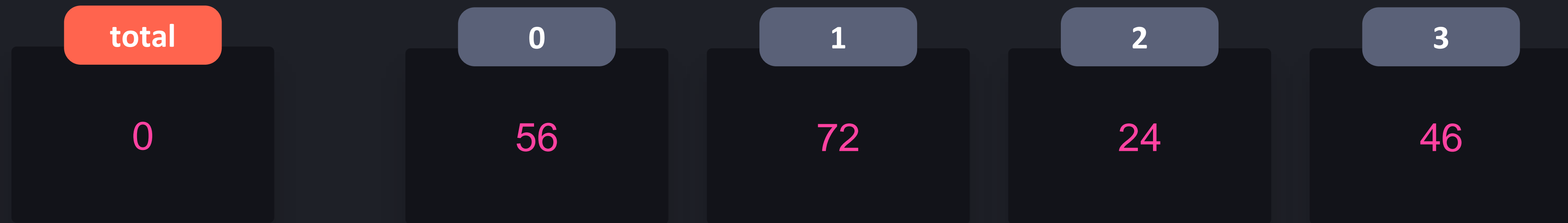


Iterating Lists



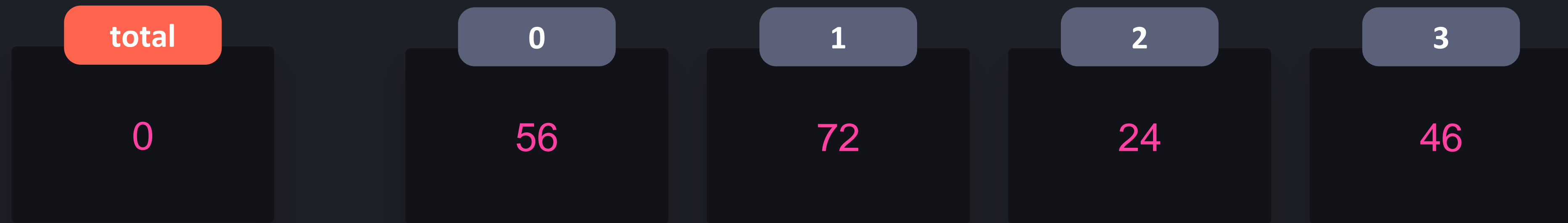
A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays a Python command to create a list named "ages" with the values 56, 72, 24, and 46.

```
>>> ages = [56, 72, 24, 46]
```



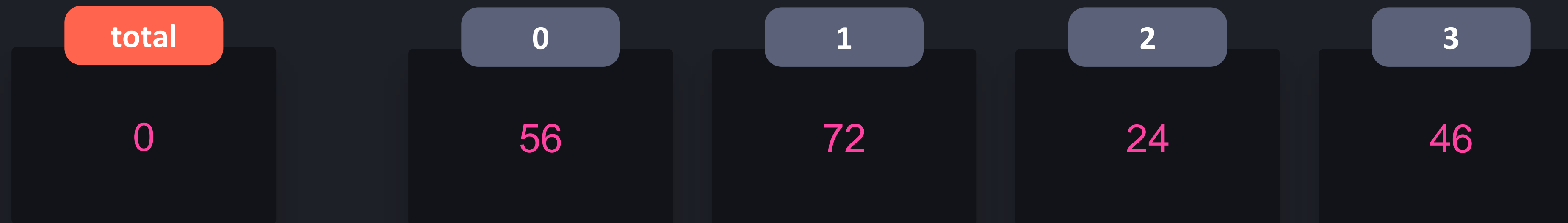
```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
```



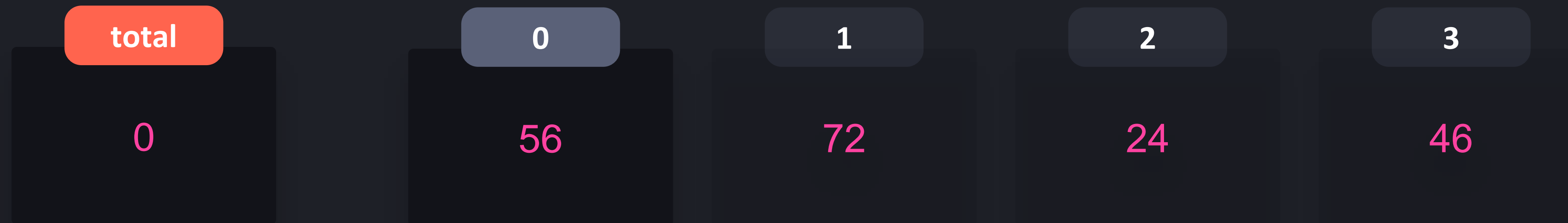
```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
>>>     total += age
```



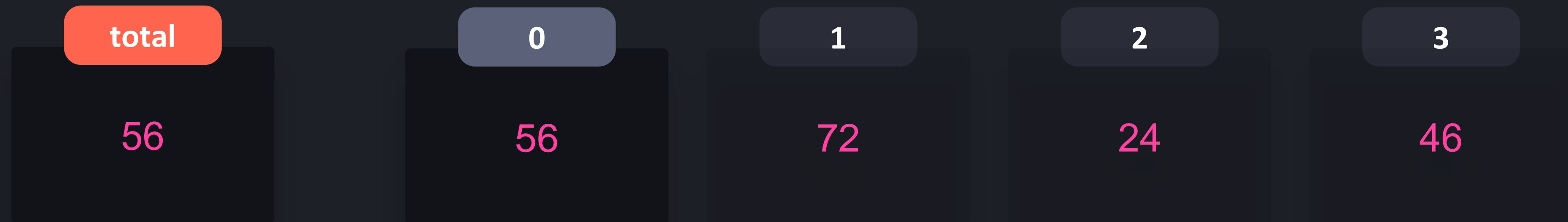
```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```



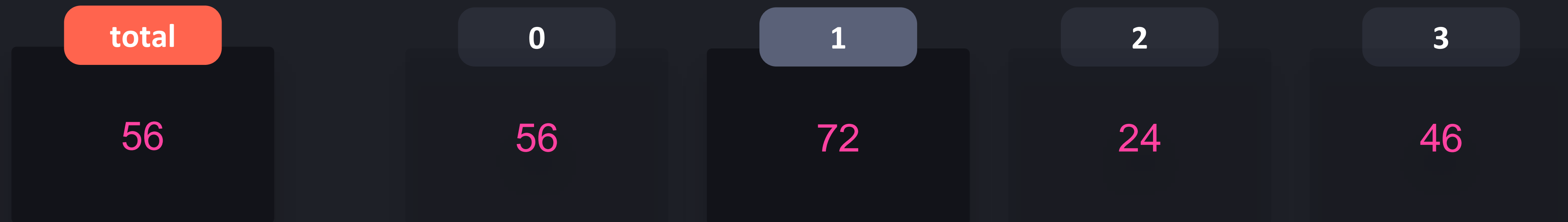
```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```



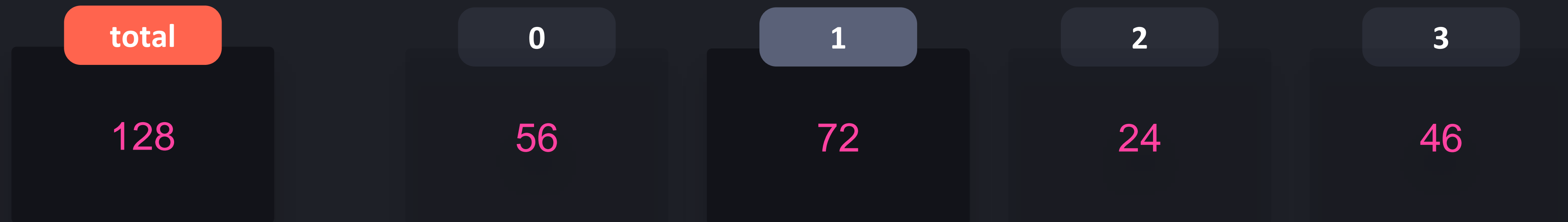
```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```

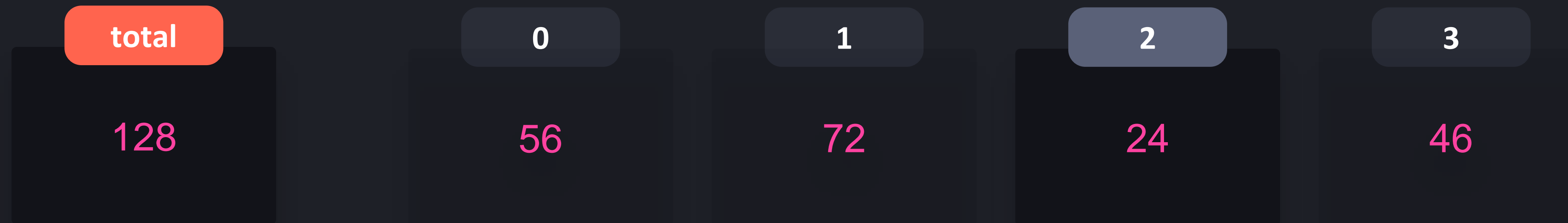
```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```



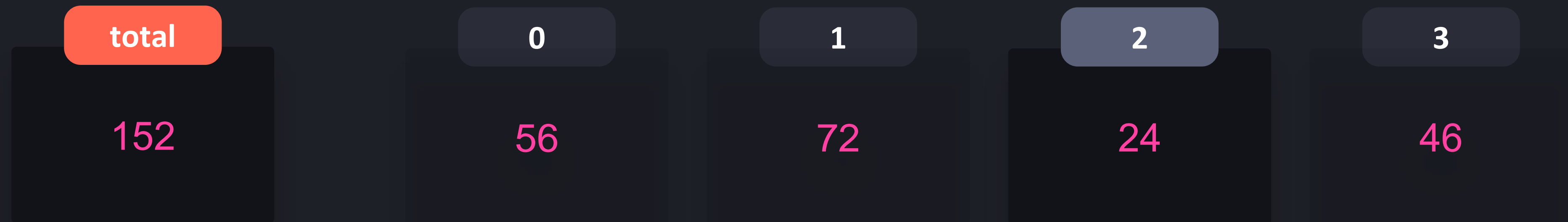
```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```



```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```



```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```

total

152

0

56

1

72

2

24

3

46



python3

```
>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```

total

198

0

56

1

72

2

24

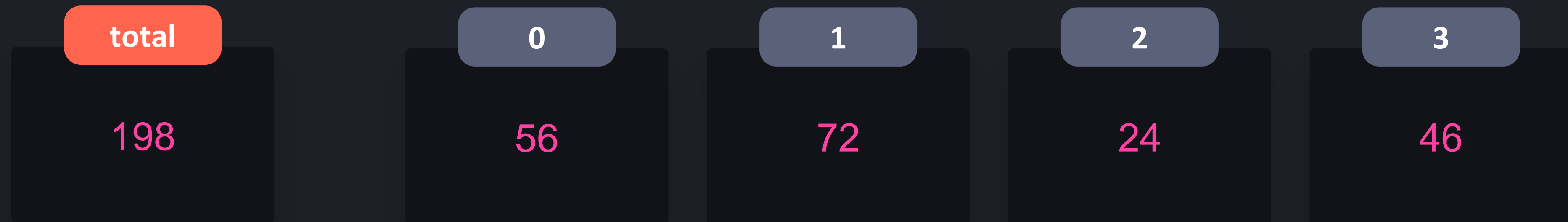
3

46



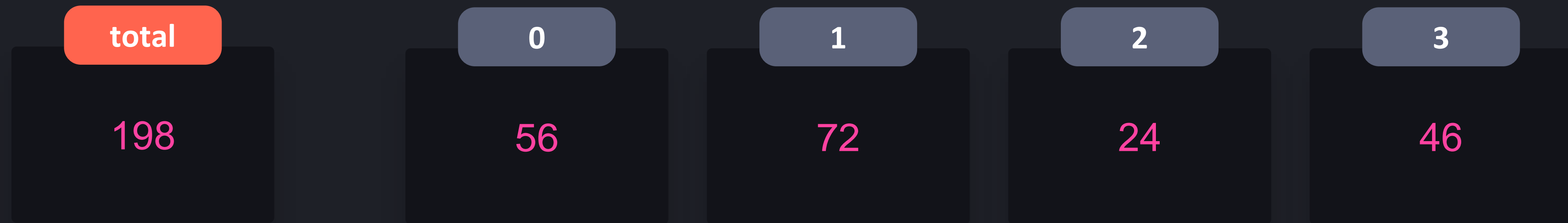
python3

```
>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```



```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```



```
python3

>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age
```


total

198

0

56

1

72

2

24

3

46

python3

```
>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
>>>     total += age
>>> average = total / len(ages)
```

total

198

0

56

1

72

2

24

3

46



python3

```
>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
    total += age

>>> average = total / len(ages)
>>> print(average)
```

total

198

0

56

1

72

2

24

3

46

python3

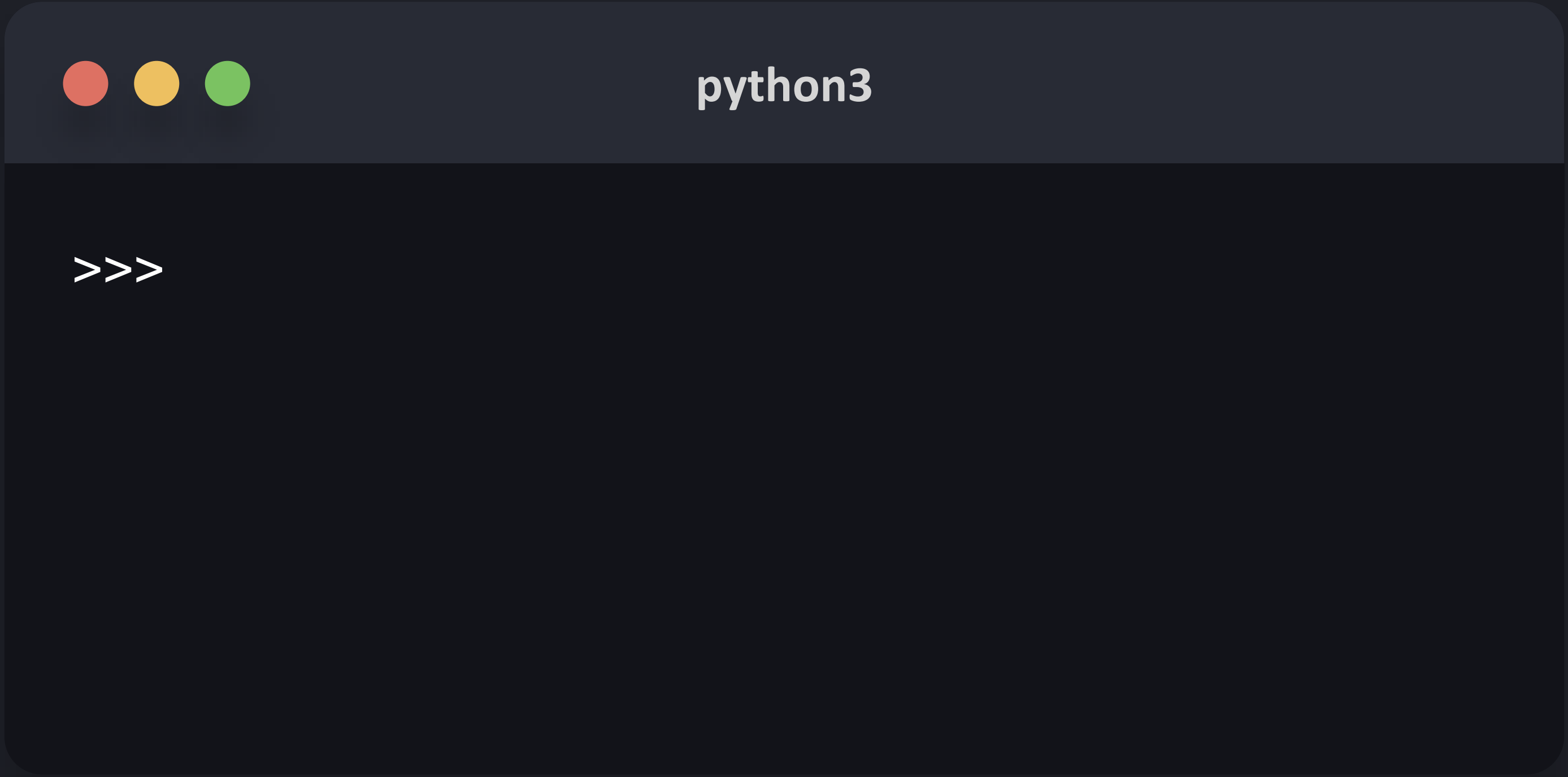
```
>>> ages = [56, 72, 24, 46]
>>> total = 0
>>> for age in ages:
>>>     total += age
>>> average = total / len(ages)
>>> print(average)
49.5
```



KodeKloud



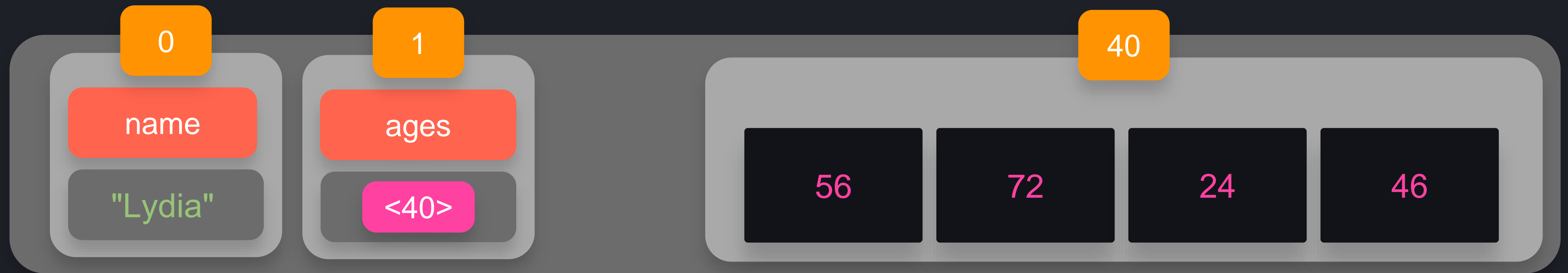
Understanding Lists





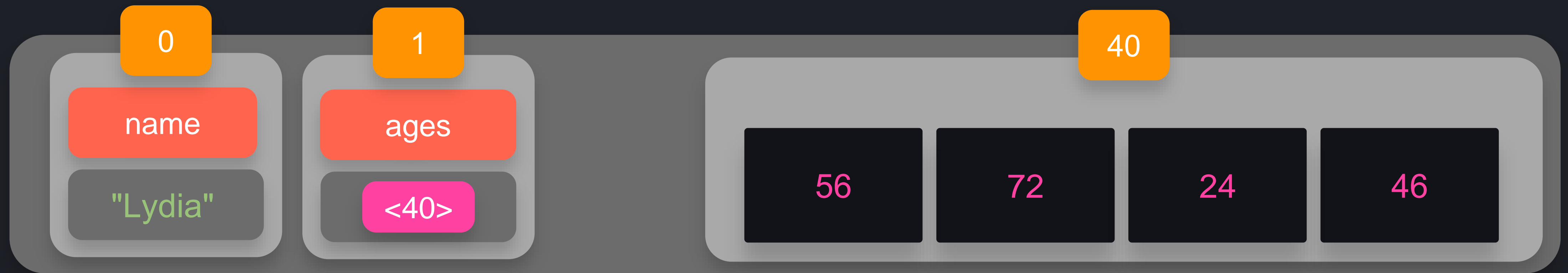
A terminal window titled 'python3' with standard macOS window controls (red, yellow, and green buttons). The terminal displays a Python prompt followed by the assignment of the string 'Lydia' to the variable 'name'.

```
>>> name = "Lydia"
```



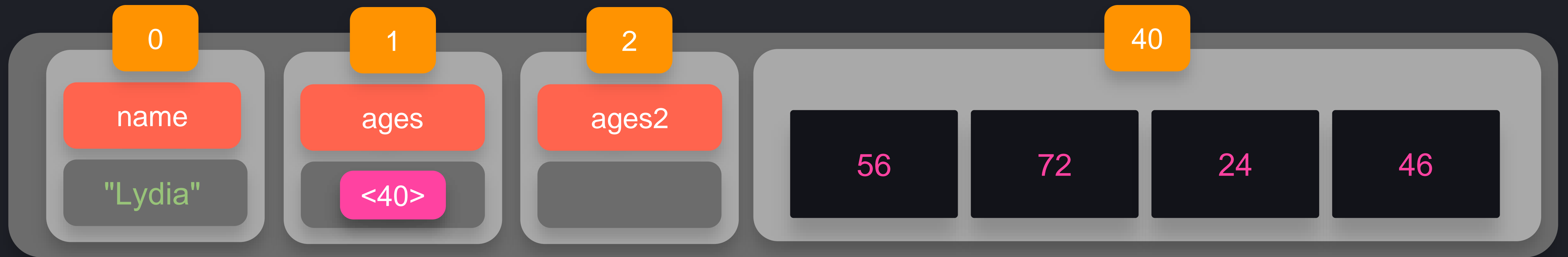
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
```

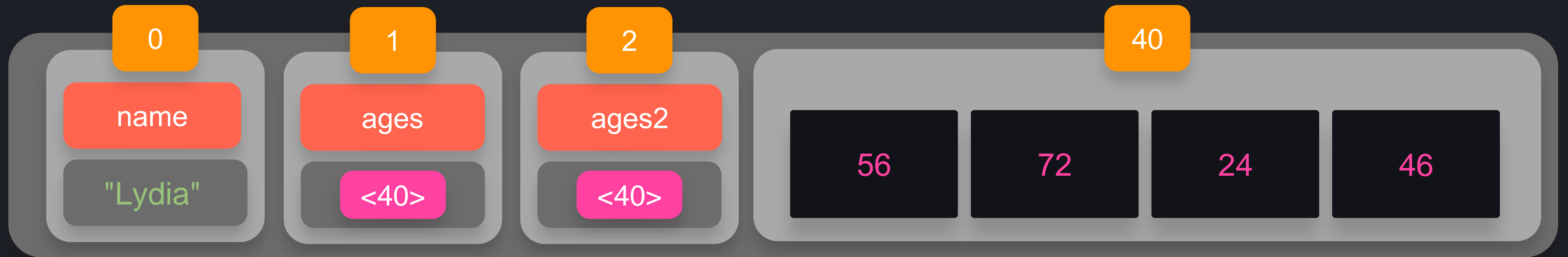
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
```



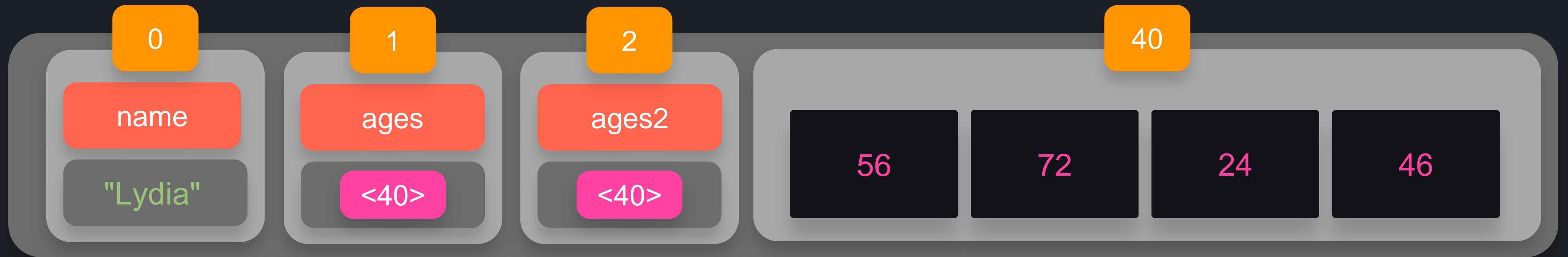
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
```



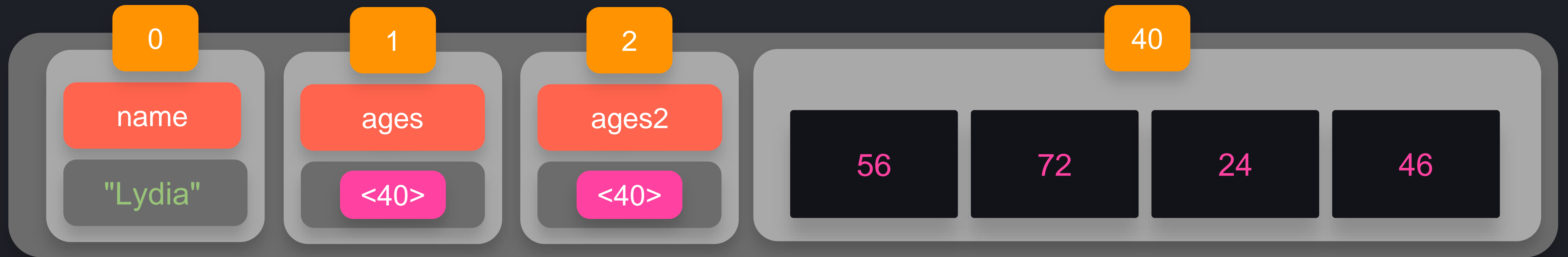
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
```



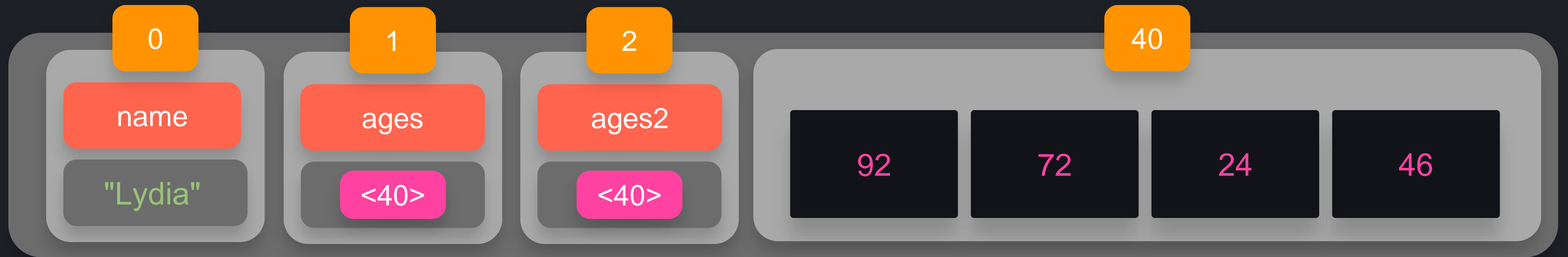
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
```



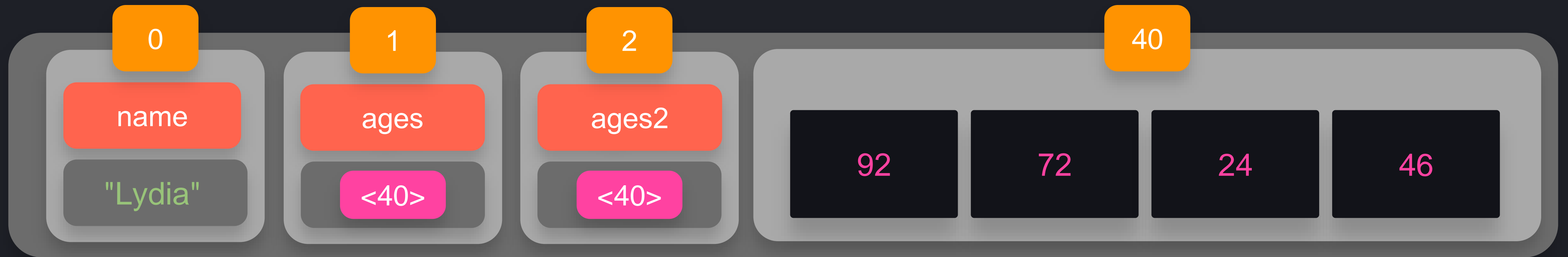
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
>>> ages[0] = 92
```



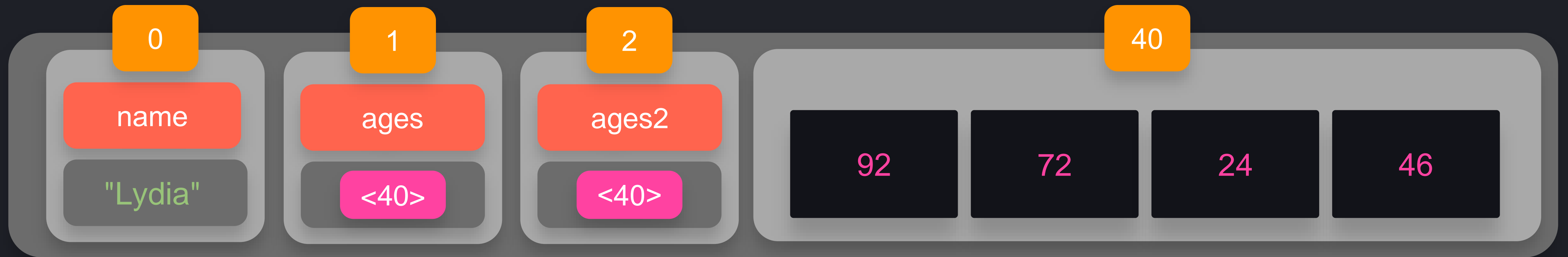
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
>>> ages[0] = 92
```



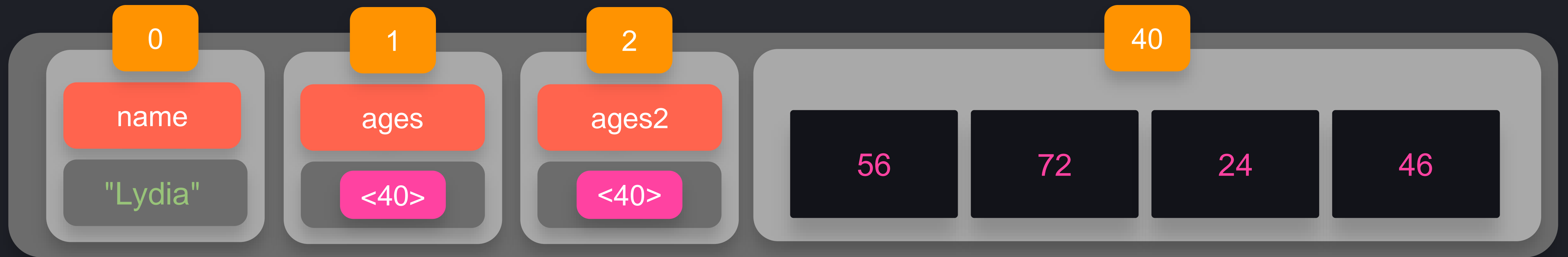
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
>>> ages[0] = 92
>>> print(ages2[0])
```



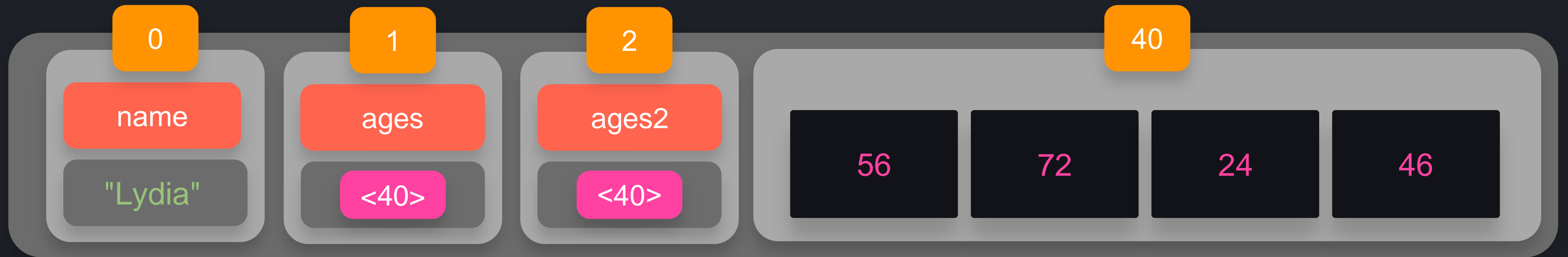
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
>>> ages[0] = 92
>>> print(ages2[0])
92
```

```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
```



```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
```

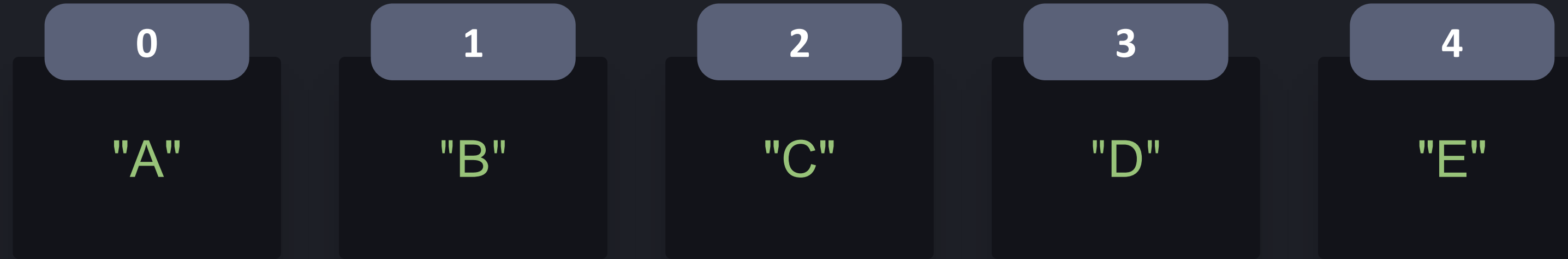


KodeKloud



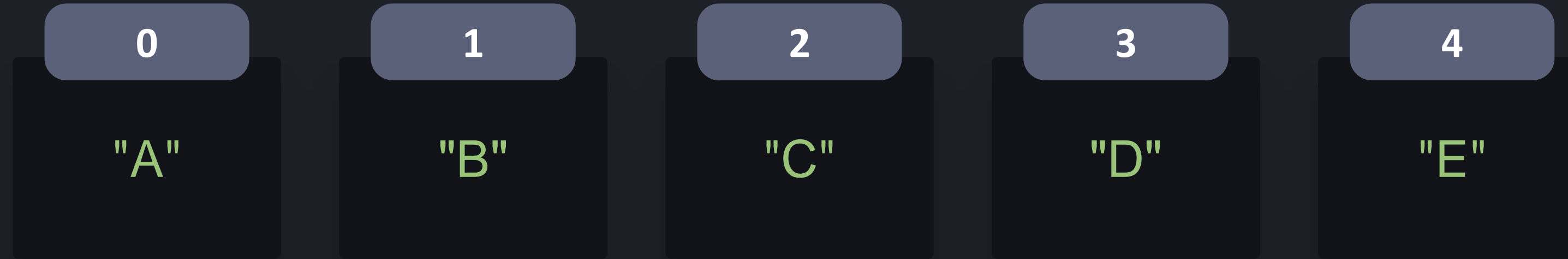
Slicing Lists

```
list[start:end]
```



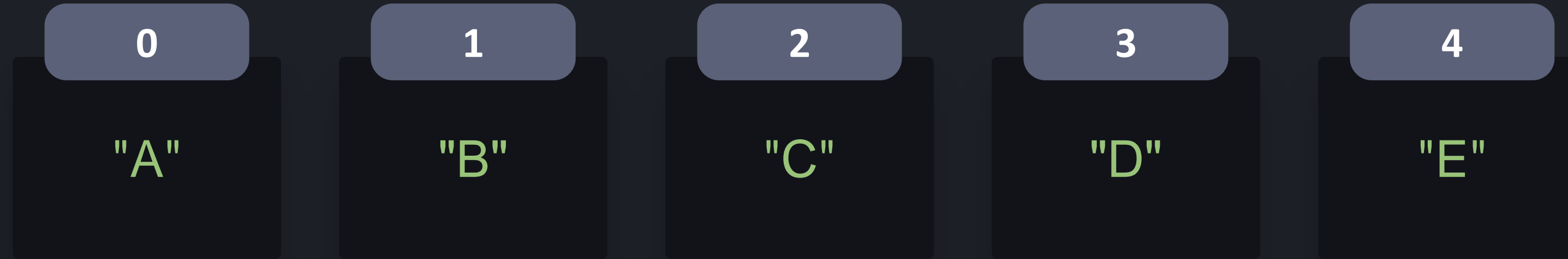
A screenshot of a Python3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text "python3" in the center. The main area of the window is dark gray and contains a single line of Python code: `>>> letters = ["A", "B", "C", "D", "E"]`. The code is color-coded: the prompt `>>>` is white, `letters` is orange, and the string elements are green.

```
>>> letters = ["A", "B", "C", "D", "E"]
```



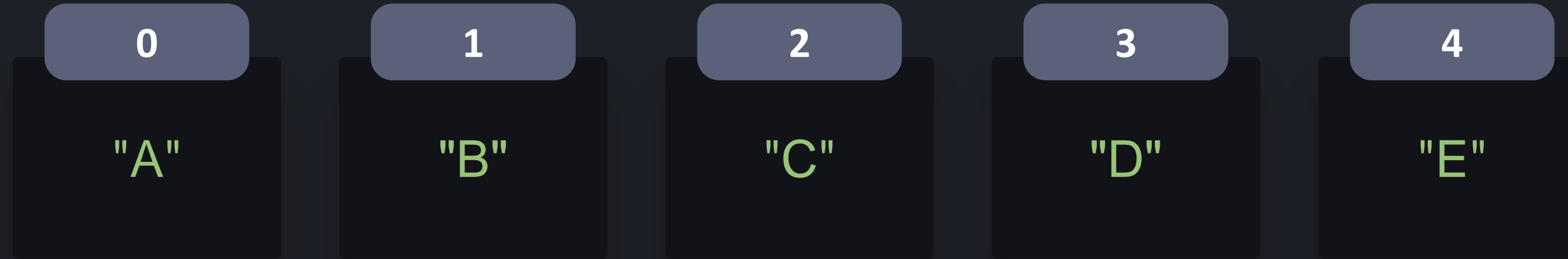
A screenshot of a Python3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text "python3" in the center. The main area of the window is dark gray and contains two lines of Python code. The first line is `>>> letters = ["A", "B", "C", "D", "E"]` and the second line is `>>> firstTwo = letters[0:2]`. The code is color-coded: the prompt `>>>` is light green, the variable names `letters` and `firstTwo` are orange, the string literals `"A"`, `"B"`, `"C"`, `"D"`, and `"E"` are light green, and the slice indices `0` and `2` are pink.

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> firstTwo = letters[0:2]
```



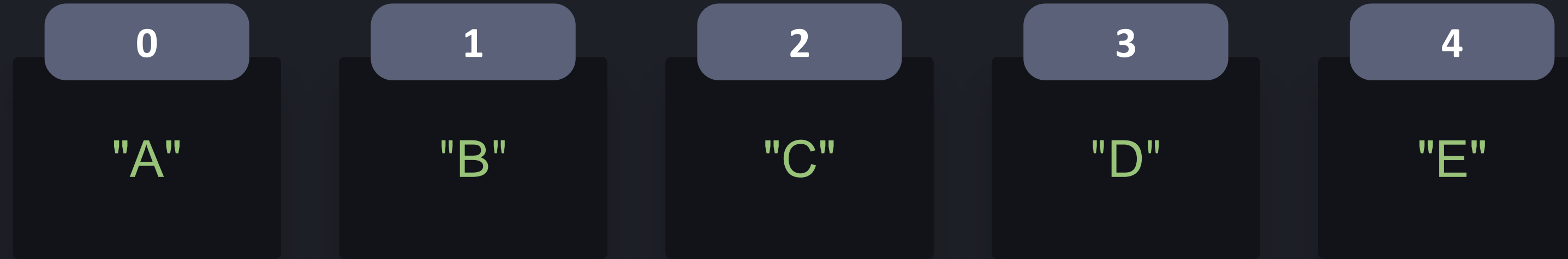
A terminal window titled "python3" with a dark background. It shows a Python interactive session where a list is created, a slice is taken, and the result is printed.

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> firstTwo = letters[0:2]
>>> print(firstTwo)
["A", "B"]
```

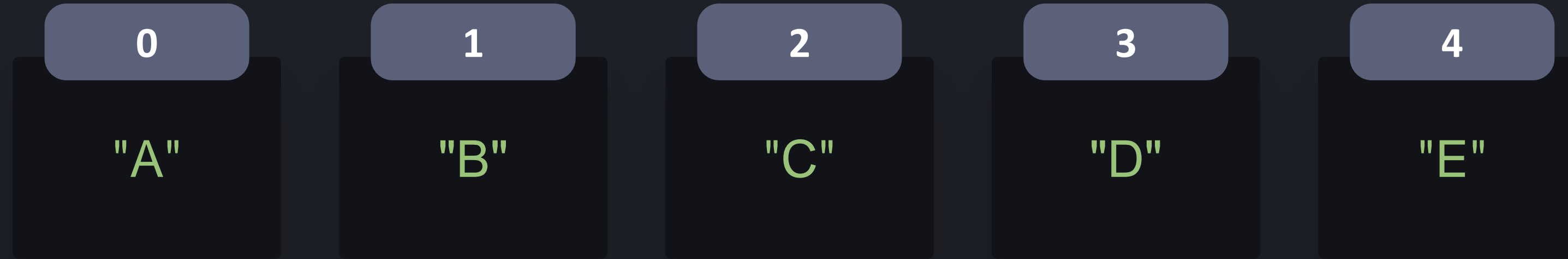
python3

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> firstTwo = letters[0:2]
>>> print(firstTwo)
["A", "B"]
>>> print(letters[1:])
```



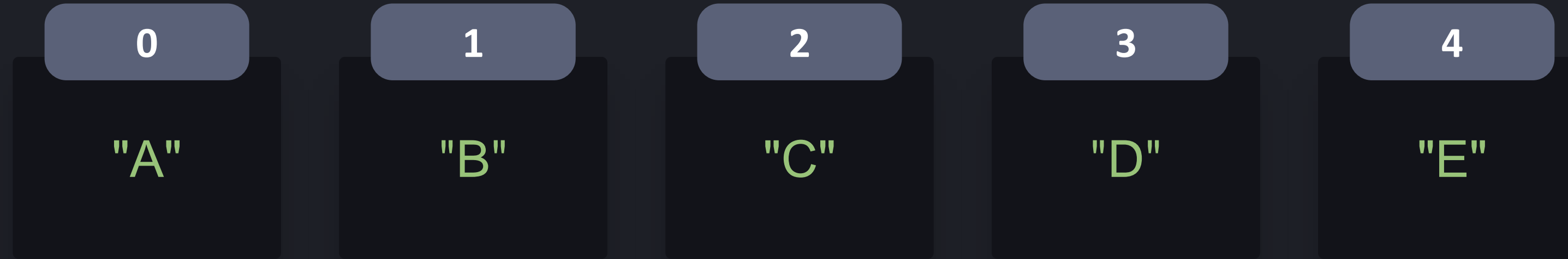
python3

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> firstTwo = letters[0:2]
>>> print(firstTwo)
["A", "B"]
>>> print(letters[1:])
["B", "C", "D", "E"]
```



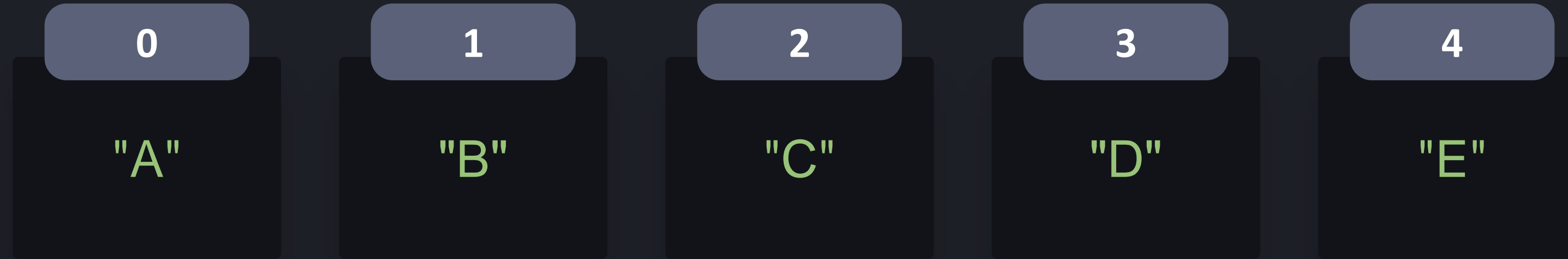
python3

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> firstTwo = letters[0:2]
>>> print(firstTwo)
["A", "B"]
>>> print(letters[1:])
["B", "C", "D", "E"]
>>> print(letters[:3])
```



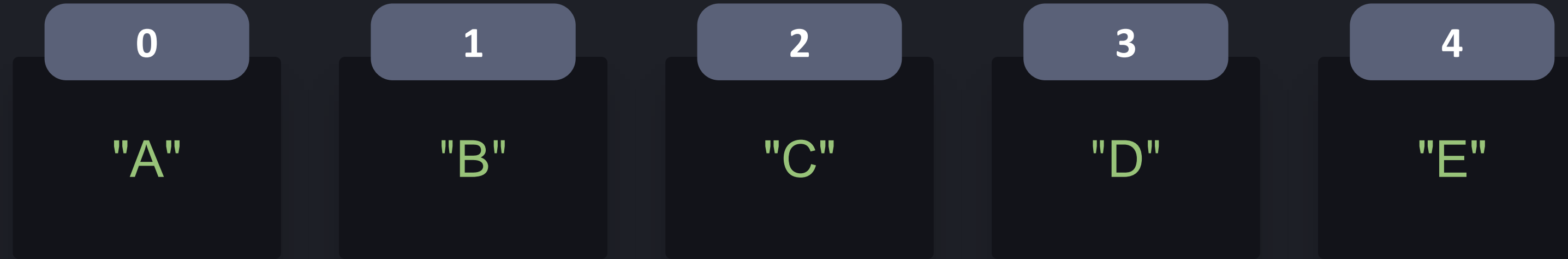
```
python3

>>> letters = ["A", "B", "C", "D", "E"]
>>> firstTwo = letters[0:2]
>>> print(firstTwo)
["A", "B"]
>>> print(letters[1:])
["B", "C", "D", "E"]
>>> print(letters[:3])
["A", "B", "C"]
```



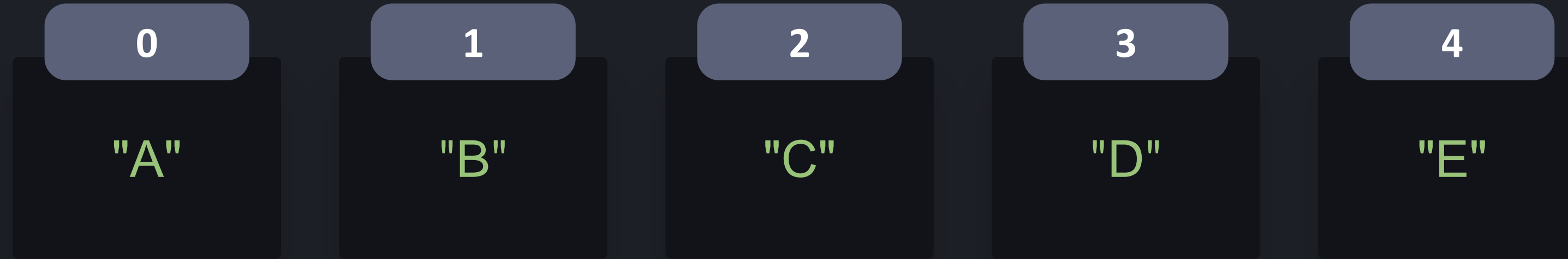
A screenshot of a Python3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text 'python3' in the center. The main area of the window is dark gray and contains a single line of Python code: `>>> letters = ["A", "B", "C", "D", "E"]`. The code is displayed in a light green monospace font, with the variable name 'letters' in orange.

```
>>> letters = ["A", "B", "C", "D", "E"]
```



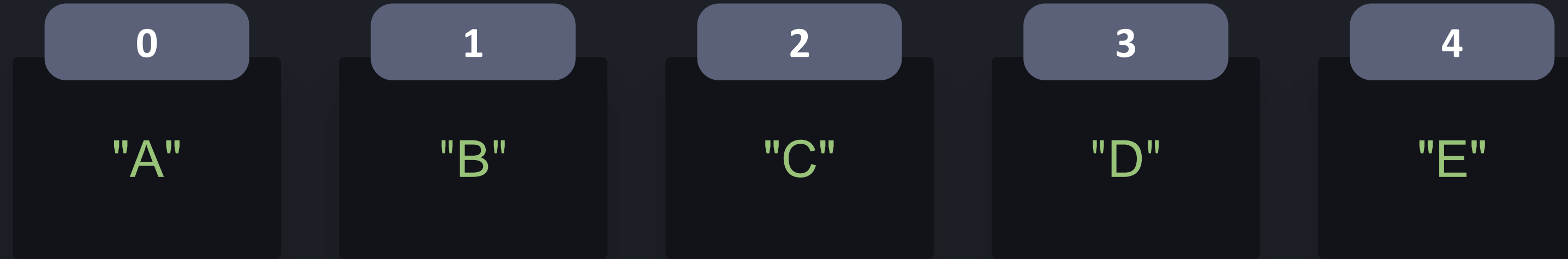
A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays the following Python code and its output:

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> print(letters[1:-1])
["B", "C", "D"]
```



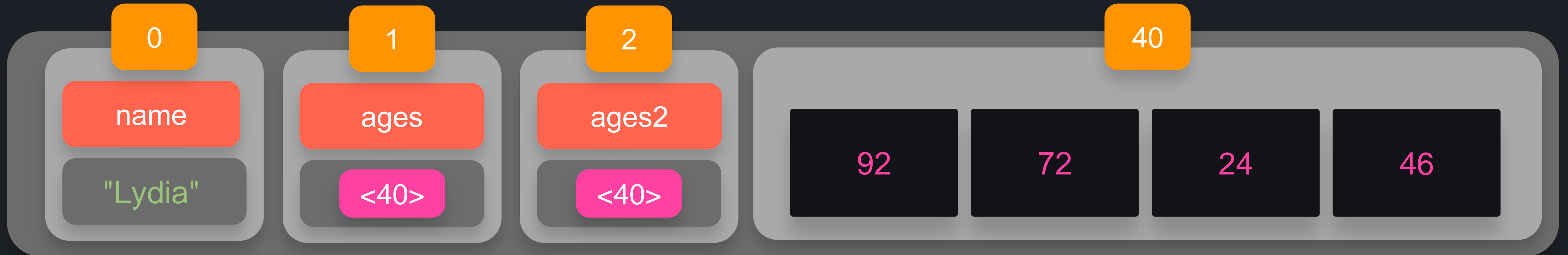
A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays two lines of Python code:

```
>>> letters = ["A", "B", "C", "D", "E"]  
>>> print(letters[:])
```



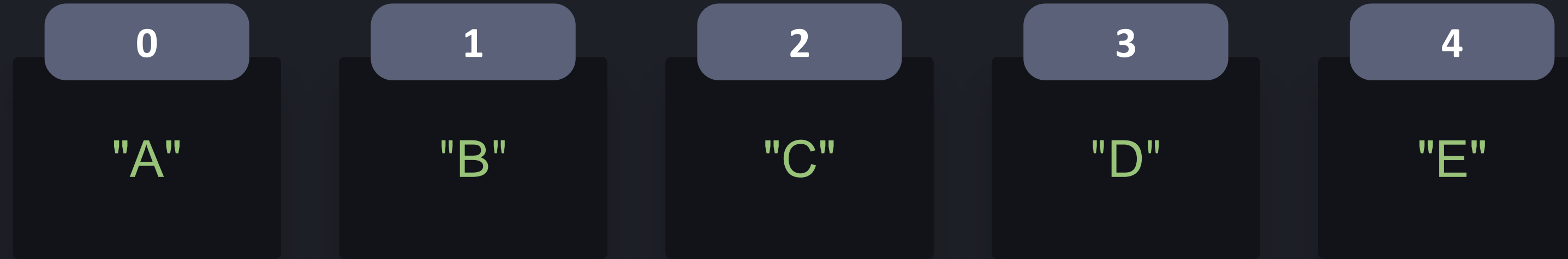
A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays the following Python code and its output:

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> print(letters[:])
["A", "B", "C", "D", "E"]
```

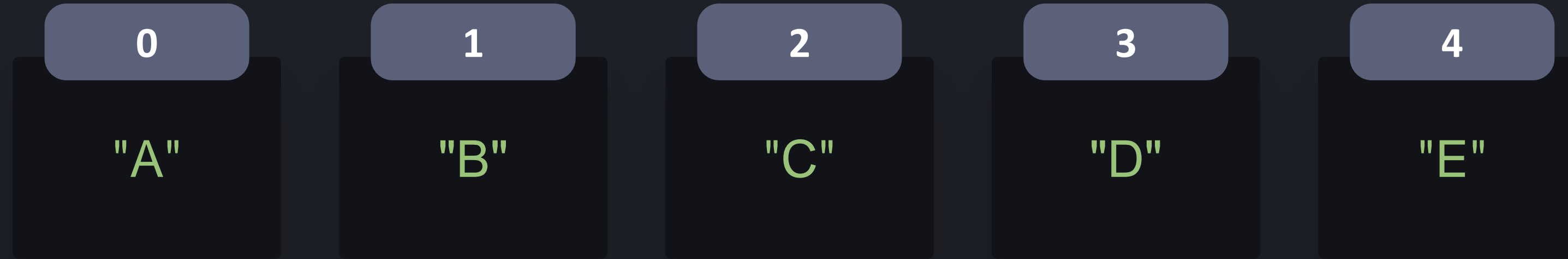
```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
>>> ages[0] = 92
>>> print(ages2[0])
92
```



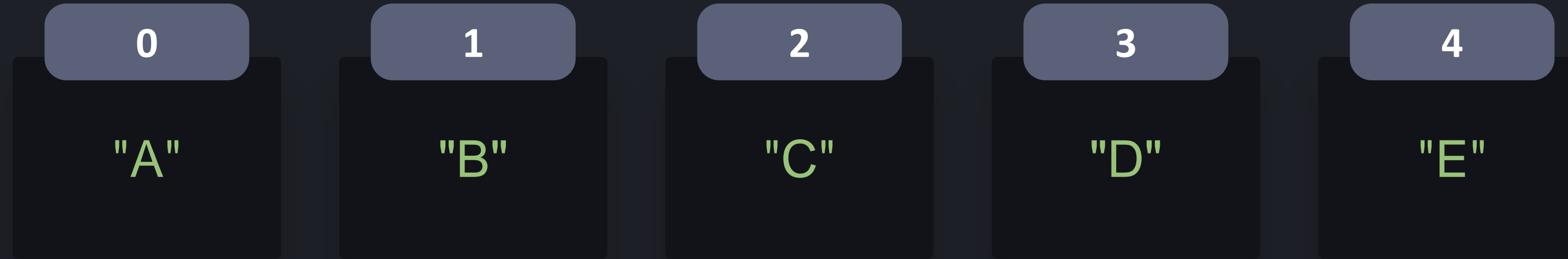
A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays the following Python code and its output:

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> print(letters[:])
["A", "B", "C", "D", "E"]
```



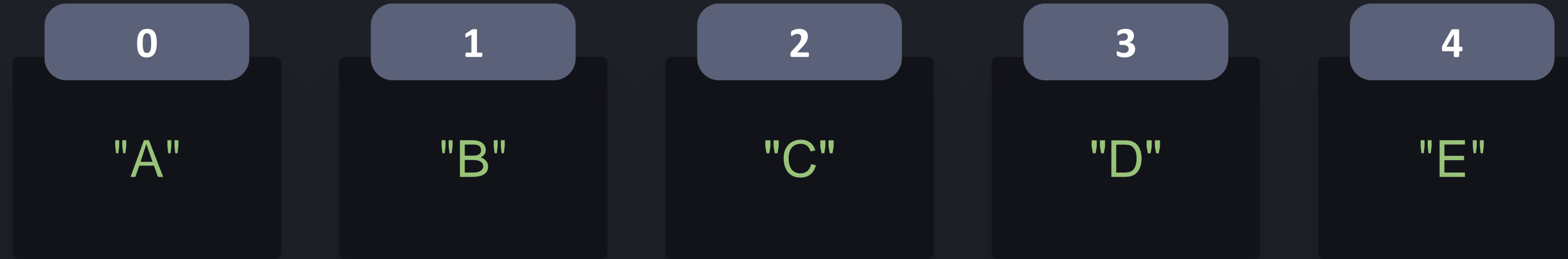
A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays the following Python code:

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> del letters[1:3]
```



A screenshot of a Python terminal window titled "python3". The window has a dark gray background and a title bar with three colored window control buttons (red, yellow, green). The terminal displays two lines of Python code:

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> del letters[1:3]
```



A terminal window titled "python3" with three colored window control buttons (red, yellow, green) on the left. The terminal displays the following Python code:

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> del letters[1:3]
```

0

"A"

1

"D"

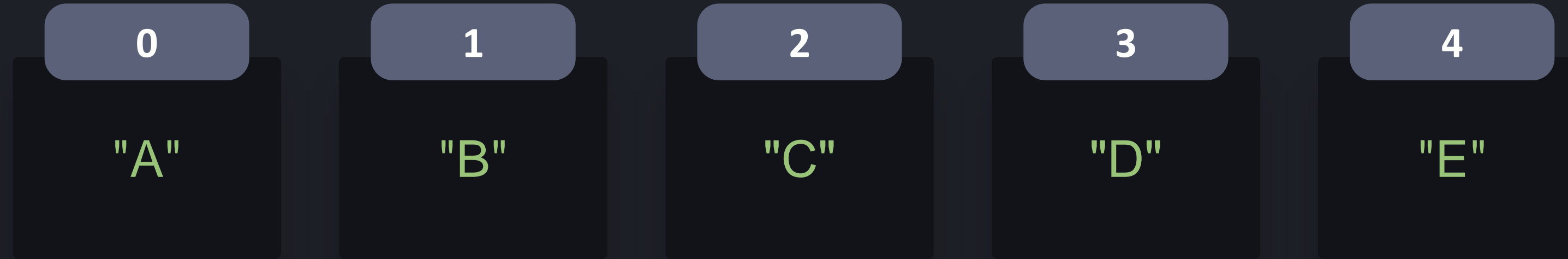
2

"E"



python3

```
>>> letters = ["A", "B", "C", "D", "E"]  
>>> del letters[1:3]
```



A screenshot of a Python3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text "python3" in the center. The main area of the window is dark gray and contains two lines of Python code. The first line is a list assignment, and the second line is a deletion statement.

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> del letters[:]
```



python3

```
>>> letters = ["A", "B", "C", "D", "E"]  
>>> del letters[:]  
>>> print(letters)
```




python3

```
>>> letters = ["A", "B", "C", "D", "E"]
```

```
>>> del letters[:]
```

```
>>> print(letters)
```

```
[]
```



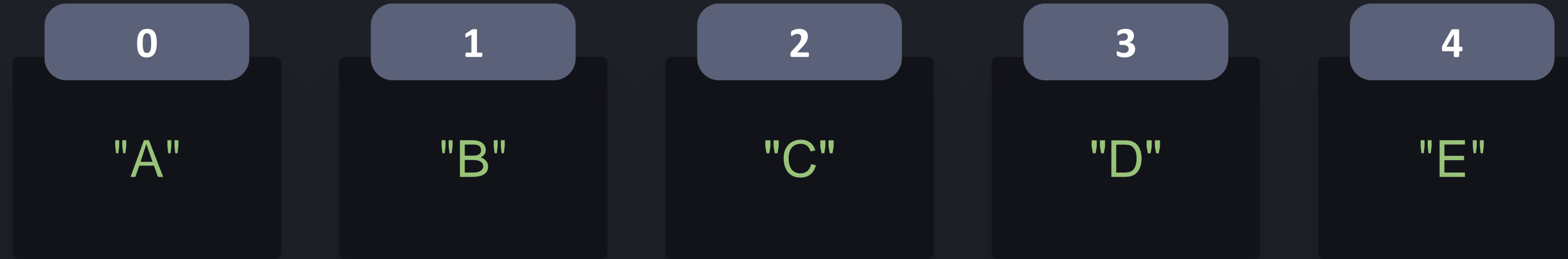
KodeKloud



Finding in Lists

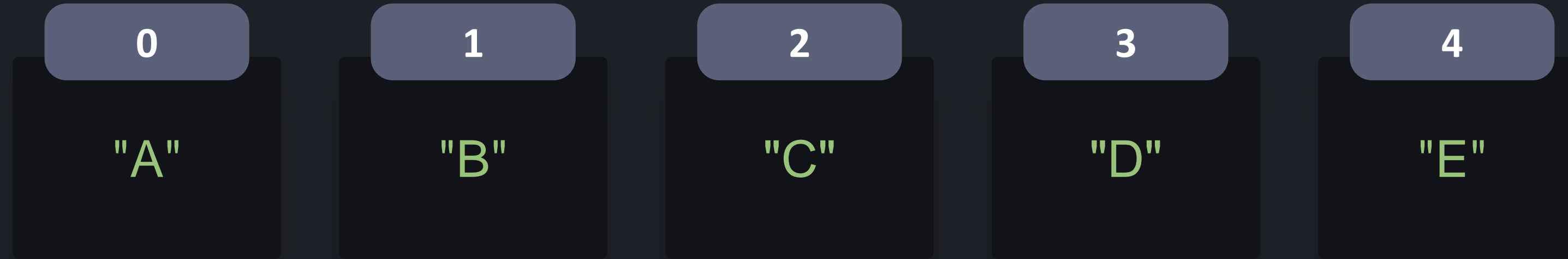
element in list

element not in list



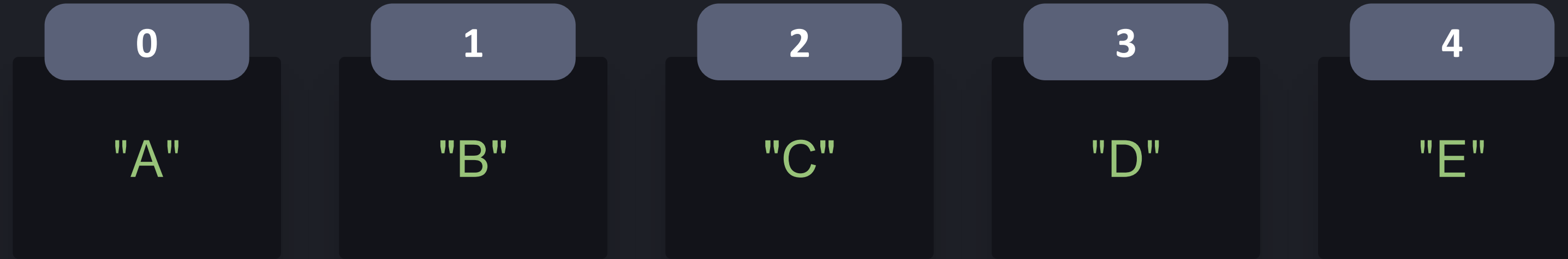
A screenshot of a Python3 terminal window. The window has a dark gray title bar with three colored window control buttons (red, yellow, green) on the left and the text "python3" in the center. The main area of the window is dark gray and contains two lines of Python code. The first line creates a list named "letters" containing the strings "A", "B", "C", "D", and "E". The second line prints the result of the expression "B" in letters, which is True.

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> print("B" in letters)
```



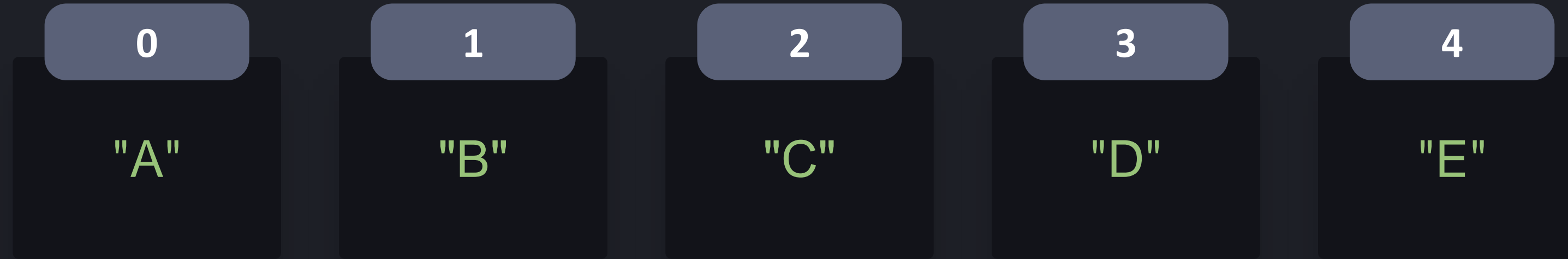
A screenshot of a Python terminal window titled "python3". The window has a dark background and a light gray title bar with three colored window control buttons (red, yellow, green). The terminal shows the following code and output:

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> print("B" in letters)
True
```



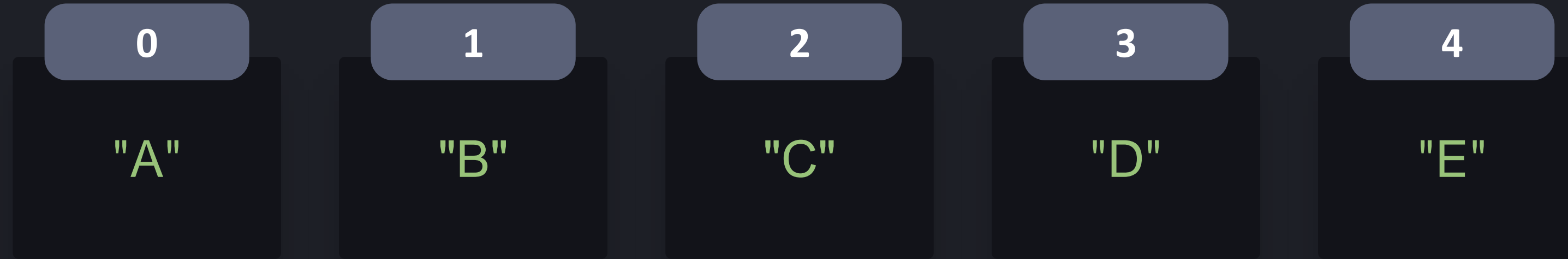
python3

```
>>> letters = ["A", "B", "C", "D", "E"]
>>> print("B" in letters)
True
>>> print("Z" in letters)
False
```



```
>>> letters = ["A", "B", "C", "D", "E"]
>>> print("B" not in letters)
False
>>> print("Z" not in letters)
True
```

A terminal window titled "python3" with standard macOS window controls (red, yellow, green buttons). It displays the following Python code and its output:



```
python3

>>> letters = ["A", "B", "C", "D", "E"]
>>> print("B" not in letters)
False
>>> print("Z" not in letters)
True
```



KodeKloud



Nested Lists - 2D

(Matrix)

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"



python3

```
>>>
```

```
classroom = [ ["Sam", "Max", "Joe", "Anne"],  
              ["Sofie", "Lisa", "Tim", "Sasha"],  
              ["Claire", "Sara", "Leo", "Kim"],  
              ["Zoe", "Guy", "Anna", "Eva"],  
              ]
```

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"

python3

```
>>>  
  
classroom = [ ["Sam", "Max", "Joe", "Anne"],  
              ["Sofie", "Lisa", "Tim", "Sasha"],  
              ["Claire", "Sara", "Leo", "Kim"],  
              ["Zoe", "Guy", "Anna", "Eva"],  
              ]
```

| | | | |
|----------|--------|--------|---------|
| "Sam" | "Max" | "Joe" | "Anne" |
| "Sofie" | "Lisa" | "Tim" | "Sasha" |
| "Claire" | "Sara" | "Leo" | "Kim" |
| "Zoe" | "Guy" | "Anna" | "Eva" |



python3

```
>>>
```

```
classroom = [ ["Sam", "Max", "Joe", "Anne"],  
              ["Sofie", "Lisa", "Tim", "Sasha"],  
              ["Claire", "Sara", "Leo", "Kim"],  
              ["Zoe", "Guy", "Anna", "Eva"],  
              ]
```

```
>>>
```

```
student = classroom[2]
```

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"



python3

```
>>>
```

```
classroom = ["Sam", "Max", "Joe", "Anne",  
             ["Sofie", "Lisa", "Tim", "Sasha"],  
             ["Claire", "Sara", "Leo", "Kim"],  
             ["Zoe", "Guy", "Anna", "Eva"],  
             ]
```

```
>>>
```

```
student = classroom[2]
```

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"



python3

```
>>>
```

```
classroom = [
    ["Sam", "Max", "Joe", "Anne"],
    ["Sofie", "Lisa", "Tim", "Sasha"],
    ["Claire", "Sara", "Leo", "Kim"],
    ["Zoe", "Guy", "Anna", "Eva"],
]
```

```
>>> student = classroom[2][1]
```

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"



python3

```
>>>
```

```
classroom = [
    ["Sam", "Max", "Joe", "Anne"],
    ["Sofie", "Lisa", "Tim", "Sasha"],
    ["Claire", "Sara", "Leo", "Kim"],
    ["Zoe", "Guy", "Anna", "Eva"],
]
```

```
>>> student = classroom[2][1]
```

```
>>> print(student)
```

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"



python3

```
>>>
```

```
classroom = [
    ["Sam", "Max", "Joe", "Anne"],
    ["Sofie", "Lisa", "Tim", "Sasha"],
    ["Claire", "Sara", "Leo", "Kim"],
    ["Zoe", "Guy", "Anna", "Eva"],
]
```

```
>>> student = classroom[2][1]
```

```
>>> print(student)
```

```
"Sara"
```

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"



KodeKloud



Nested Lists - 3D

(Cube)

"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

"Guy"

"Anna"

"Eva"



"Sam"

"Max"

"Joe"

"Anne"

"Sofie"

"Lisa"

"Tim"

"Sasha"

"Claire"

"Sara"

"Leo"

"Kim"

"Zoe"

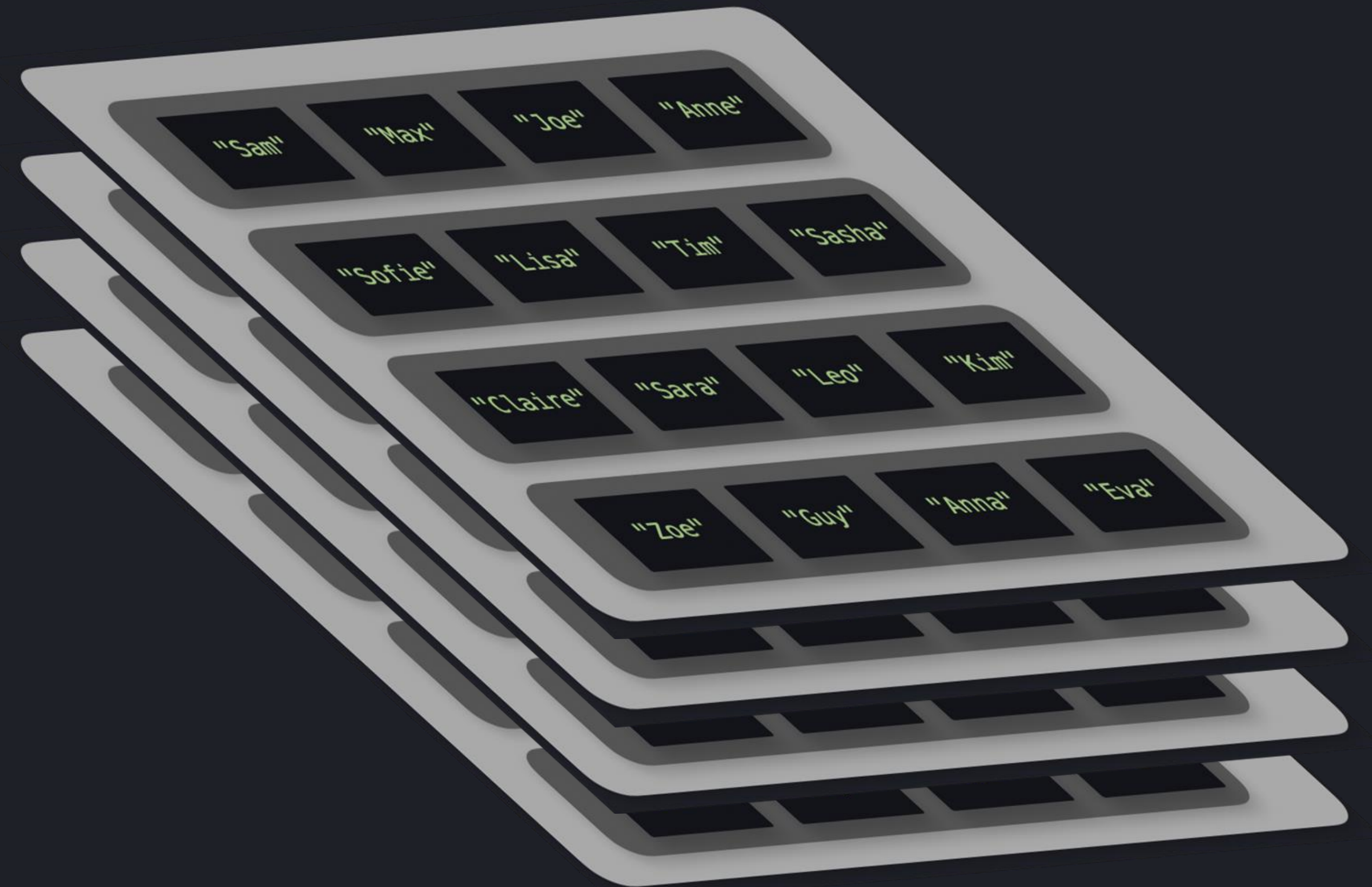
"Guy"

"Anna"

"Eva"

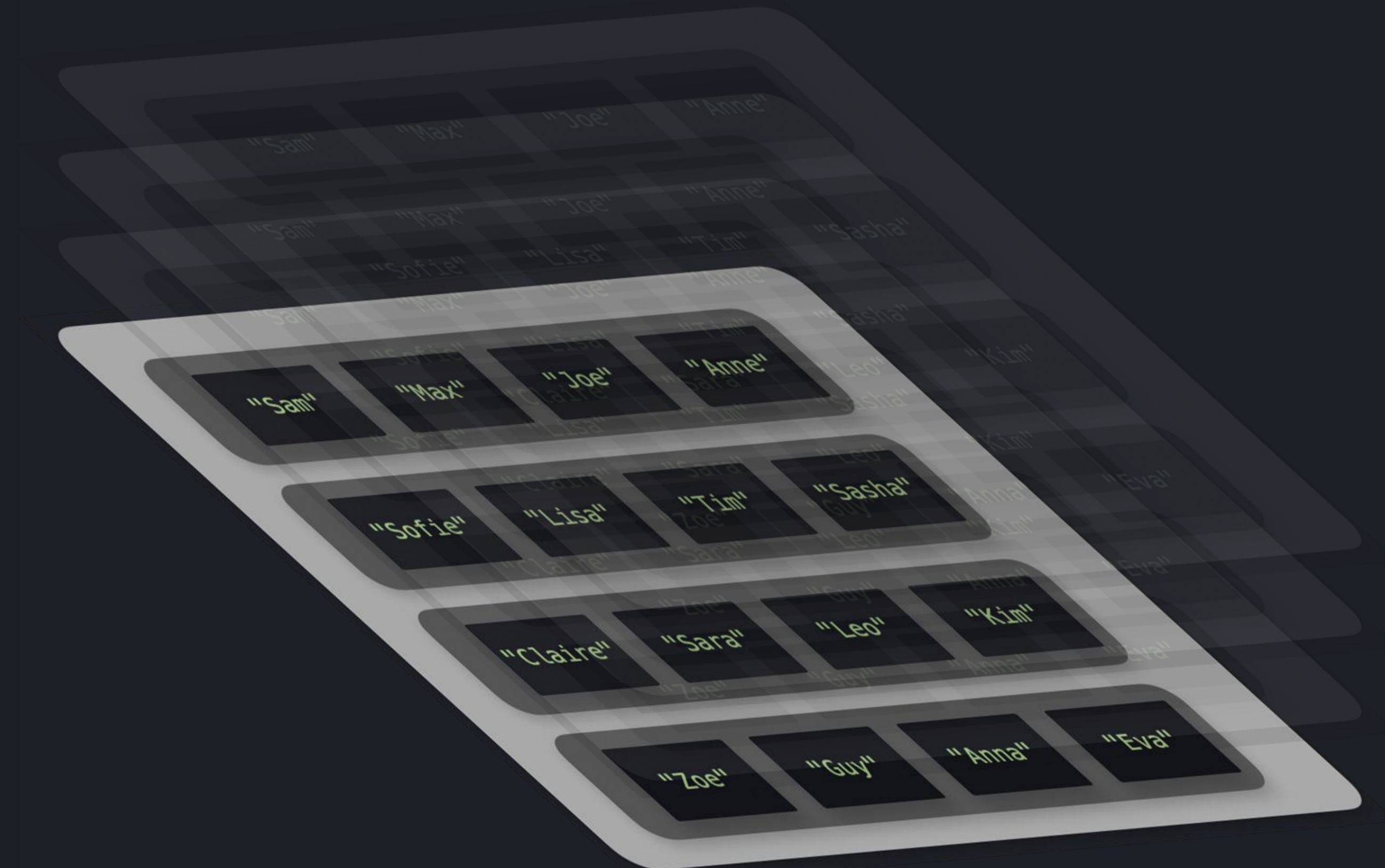
python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```



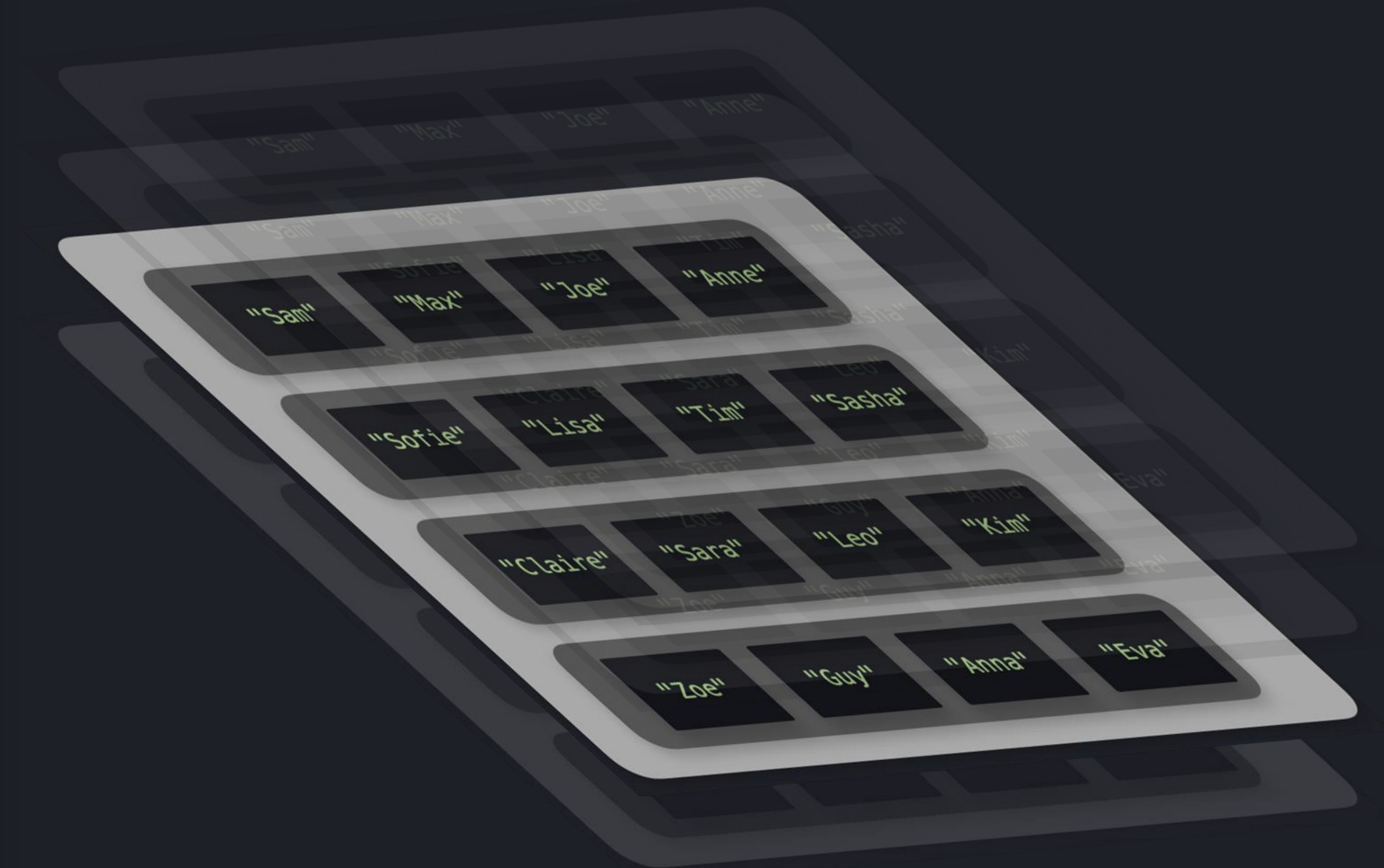
python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```



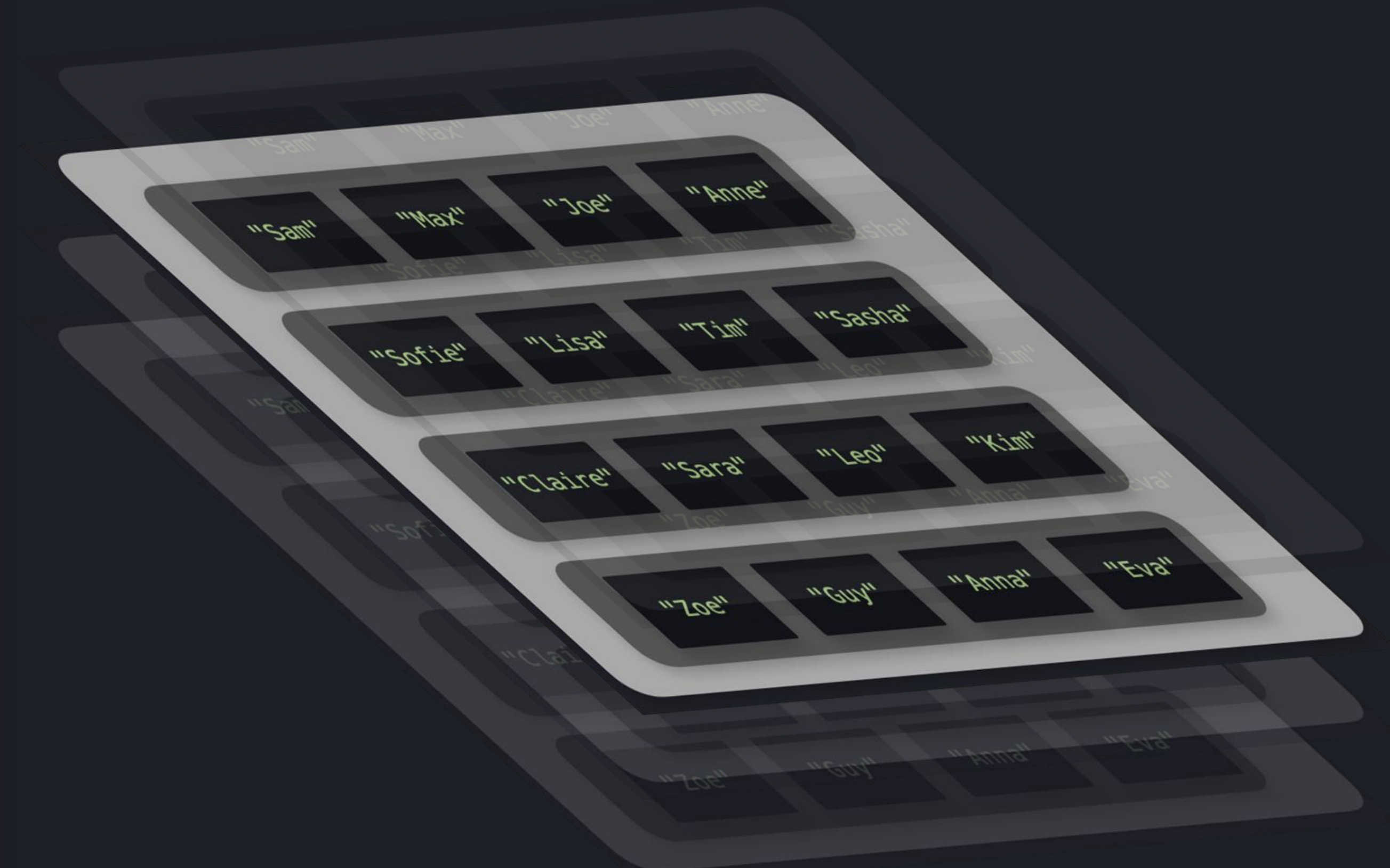
python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```



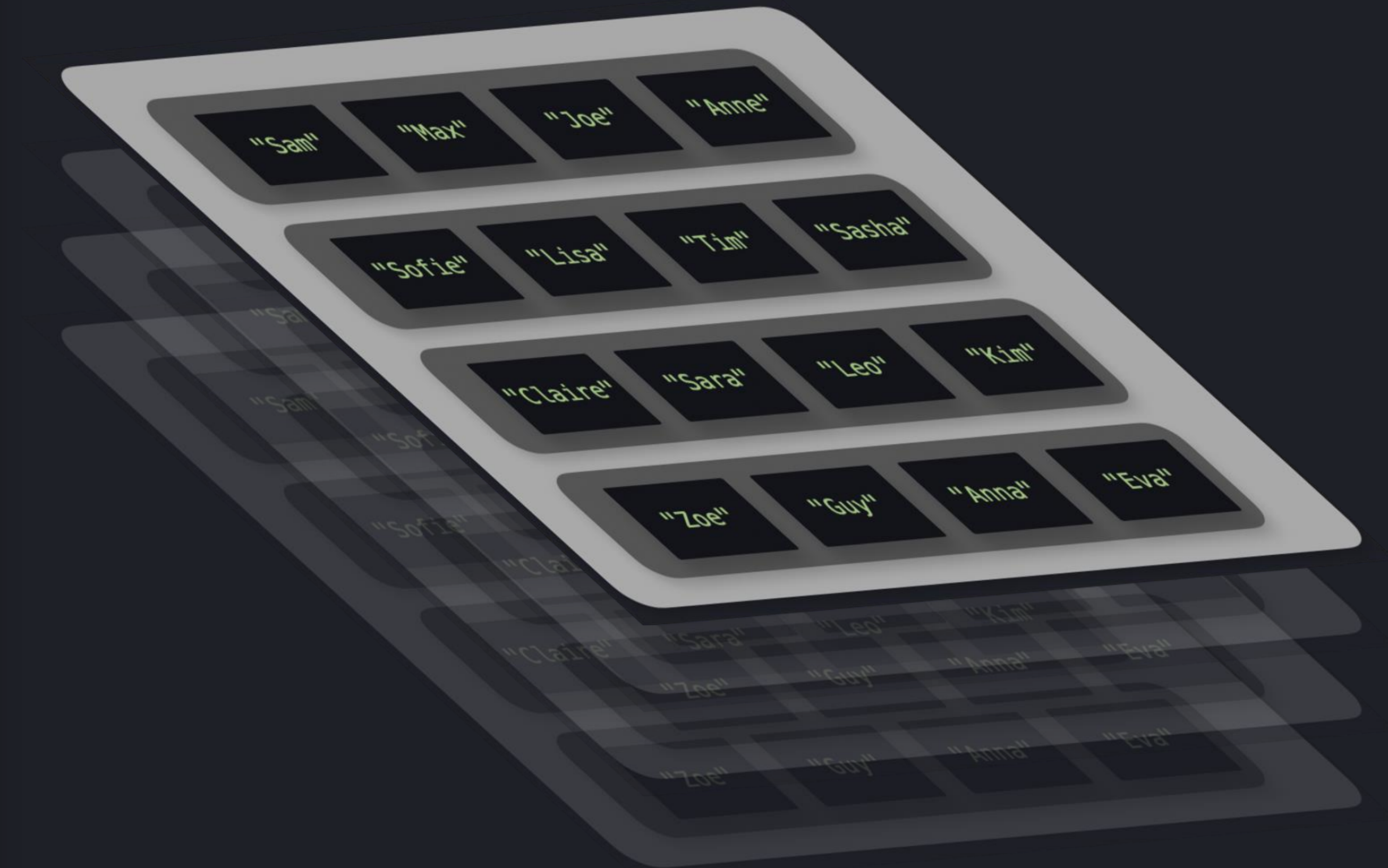
python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```



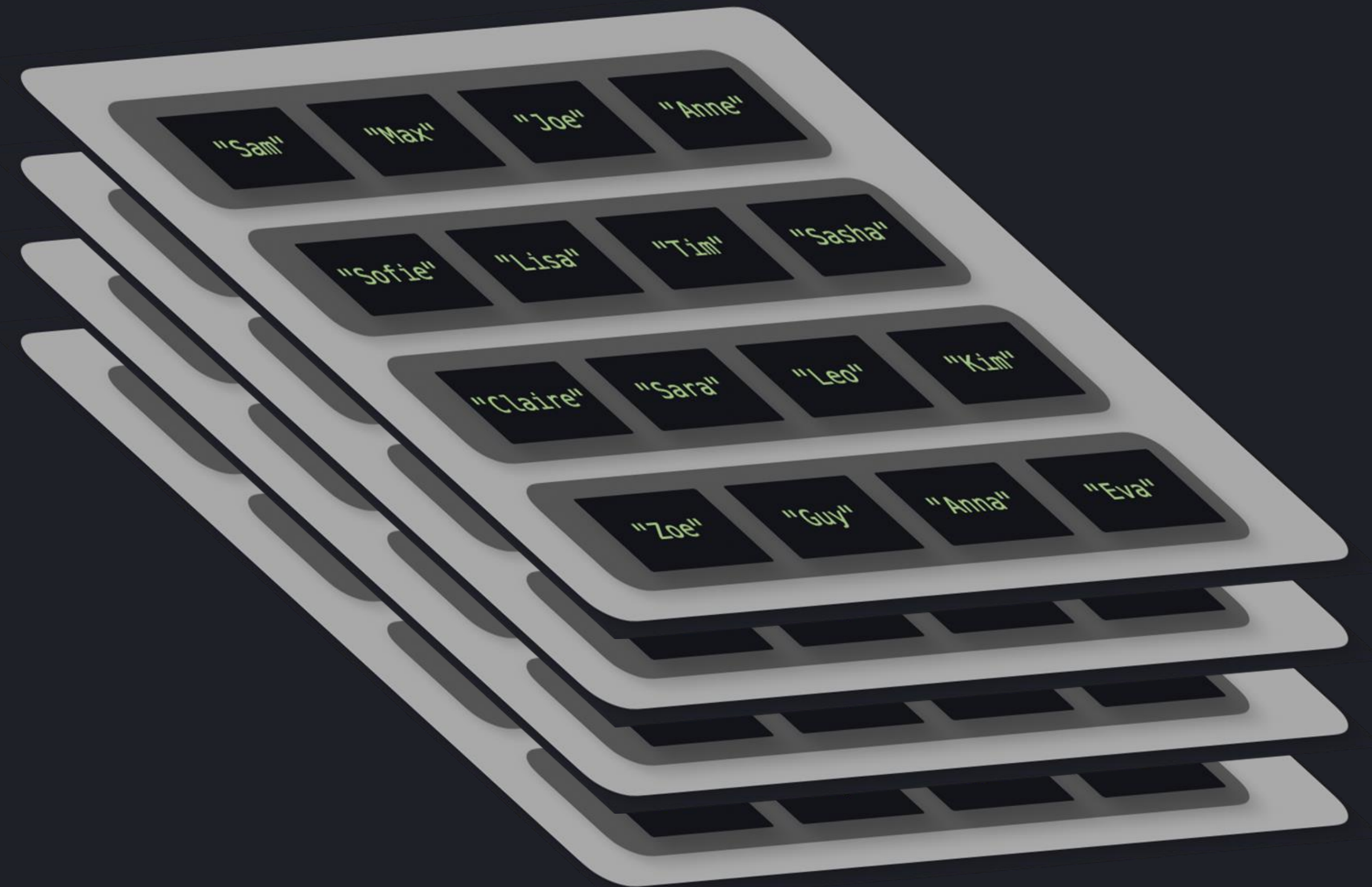
python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```



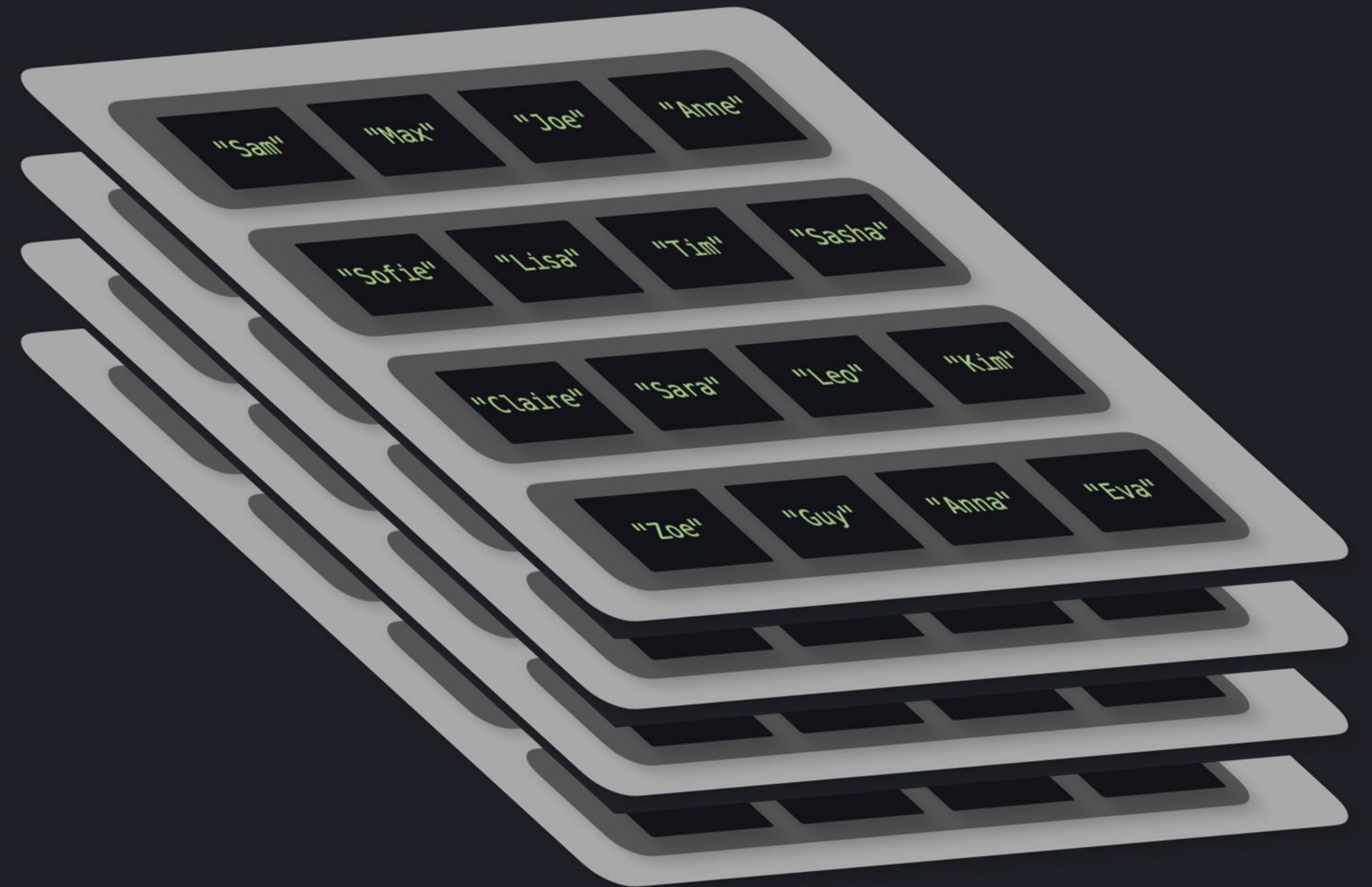
python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```



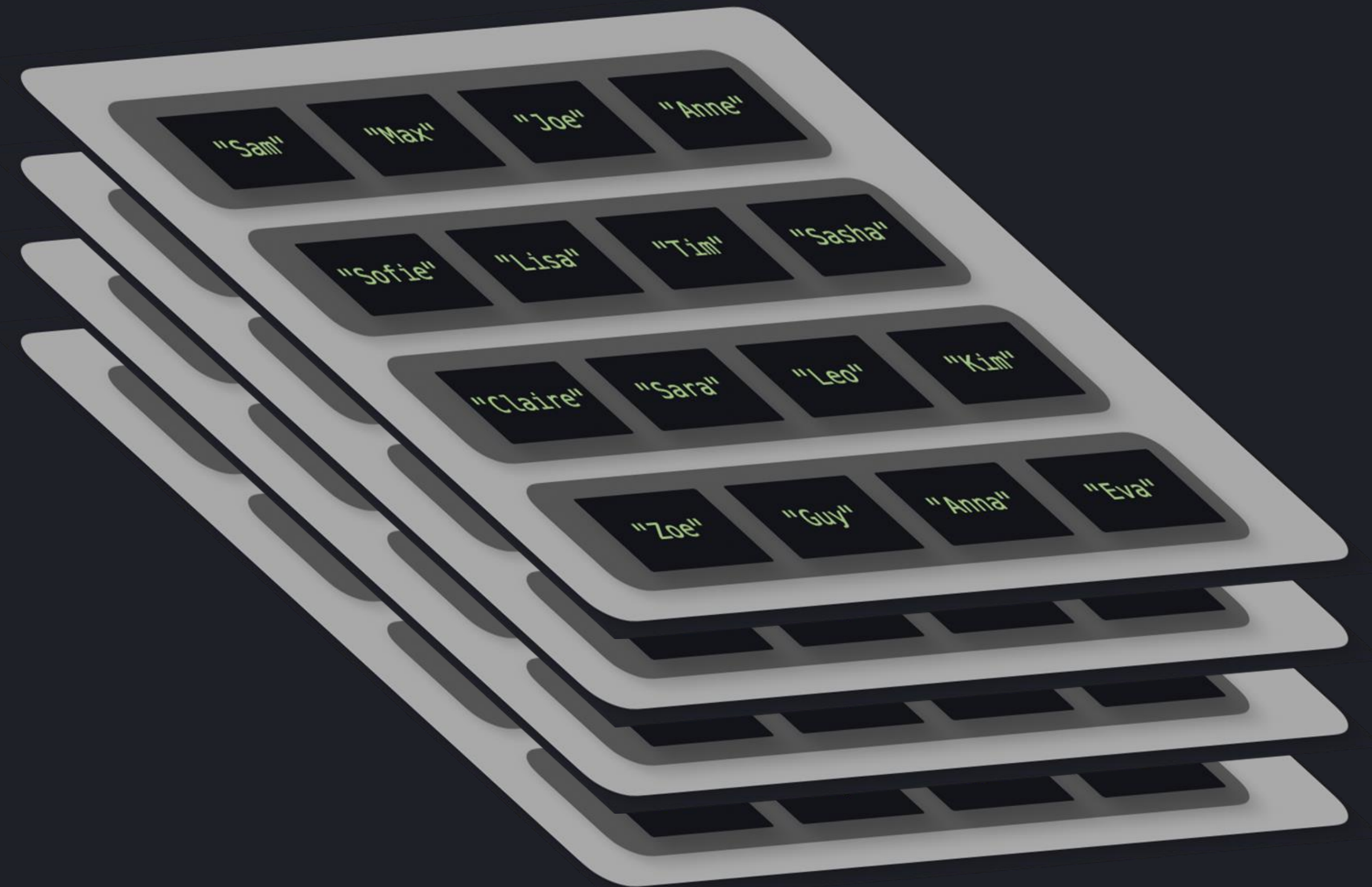

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

```
>>>
```



```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

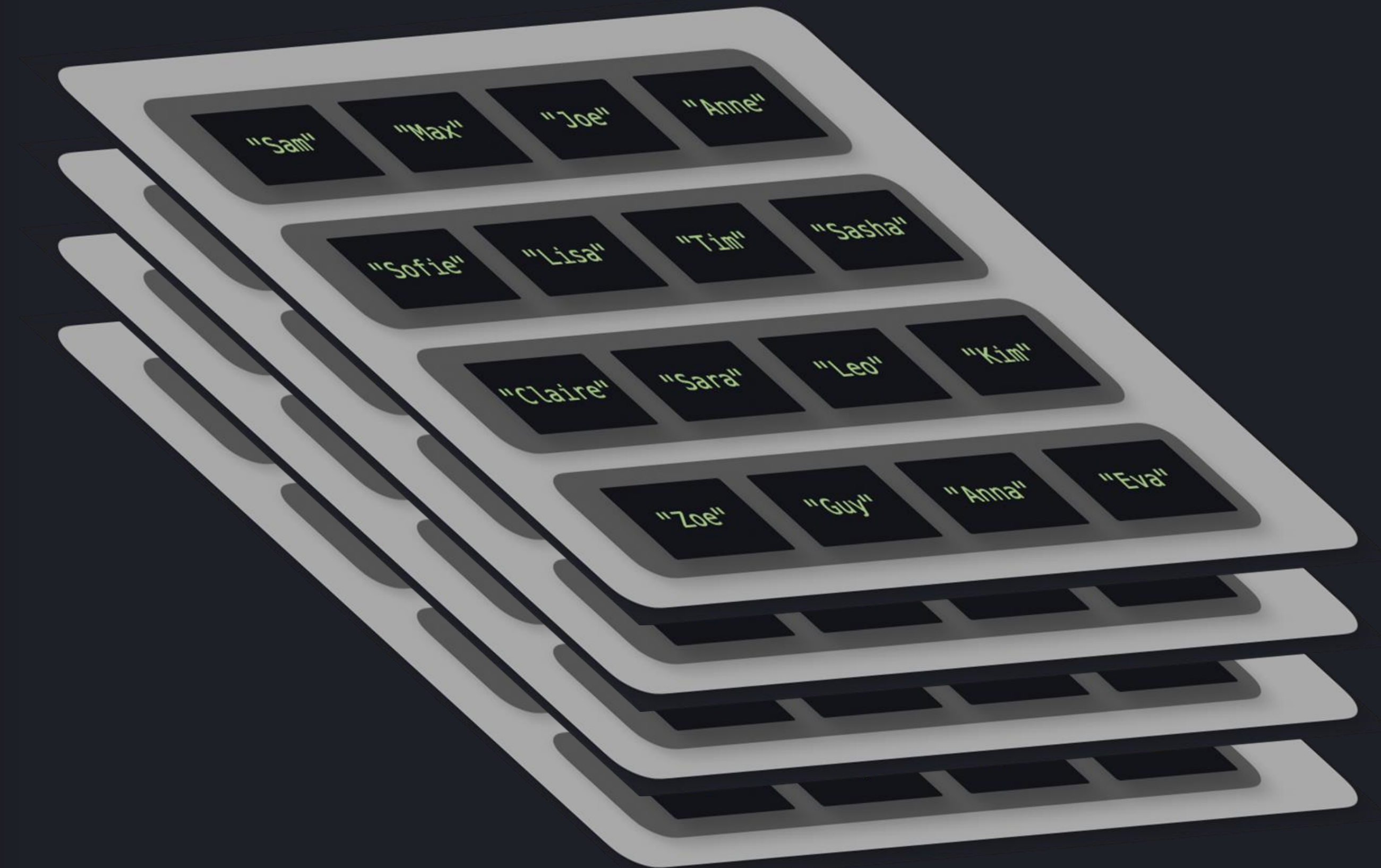
```
>>>
```



python3

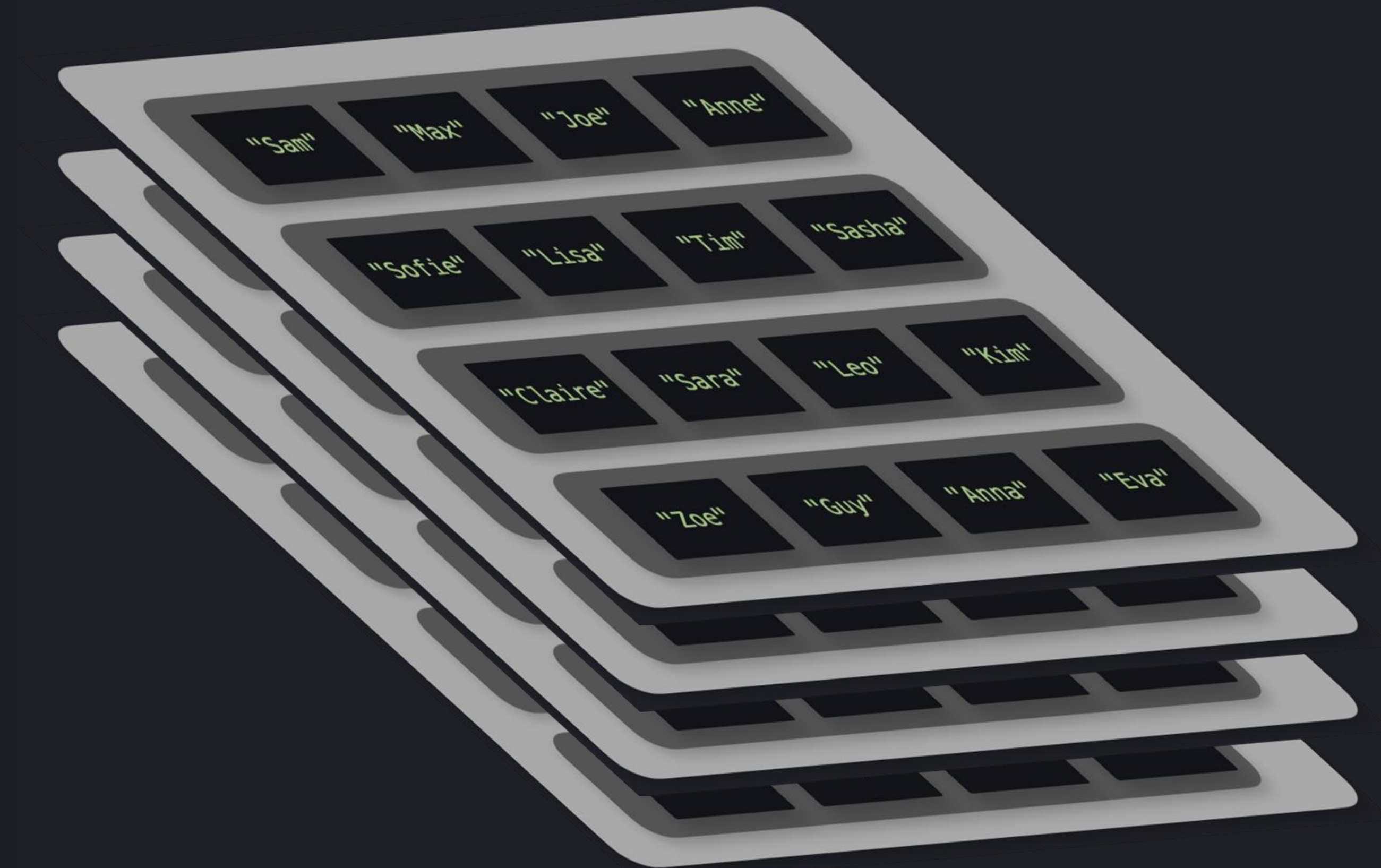
```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

```
>>> student = school[1]
```



```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

```
>>> student = school[1]
```

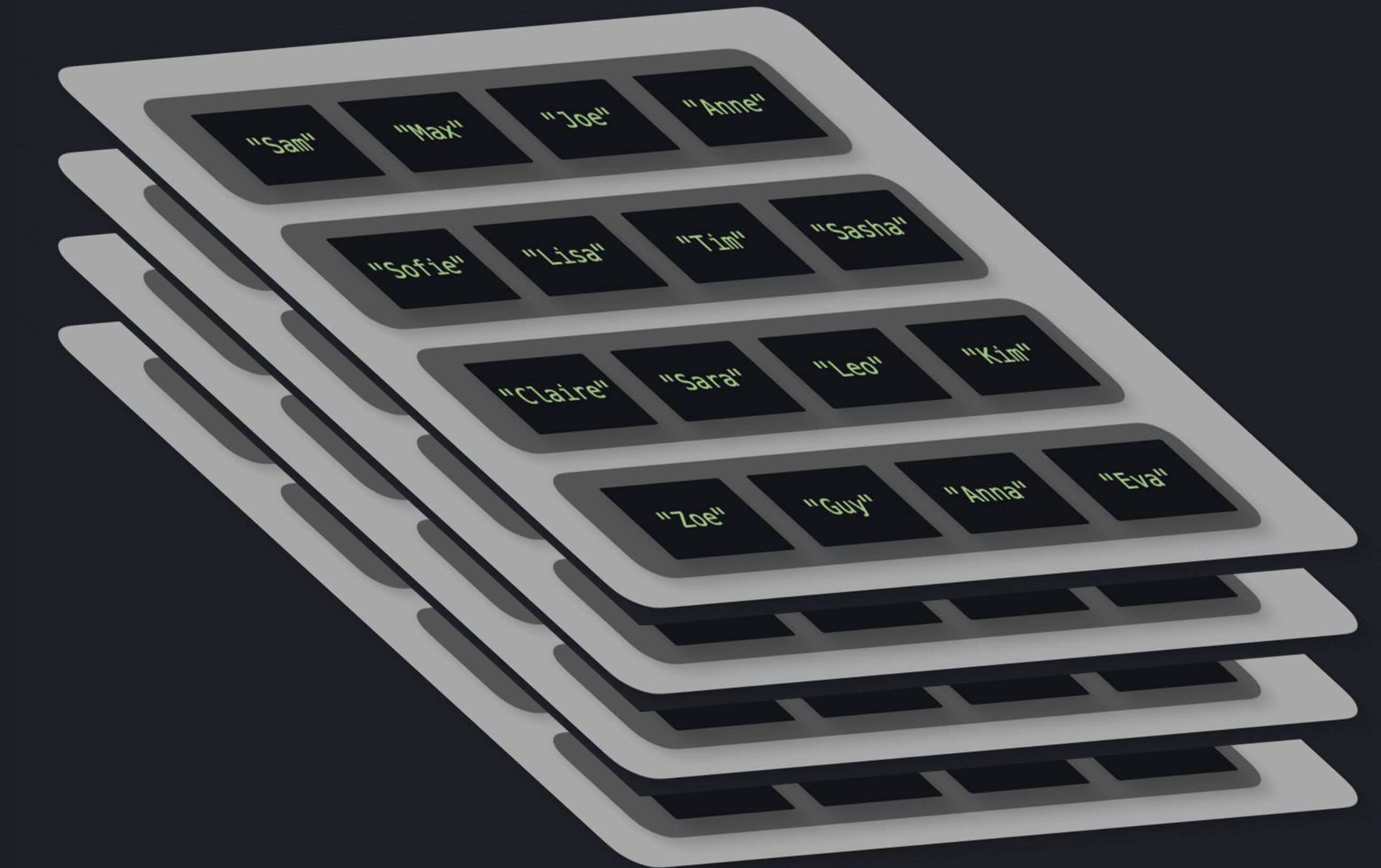




python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

```
>>> student = school[1] [2]
```

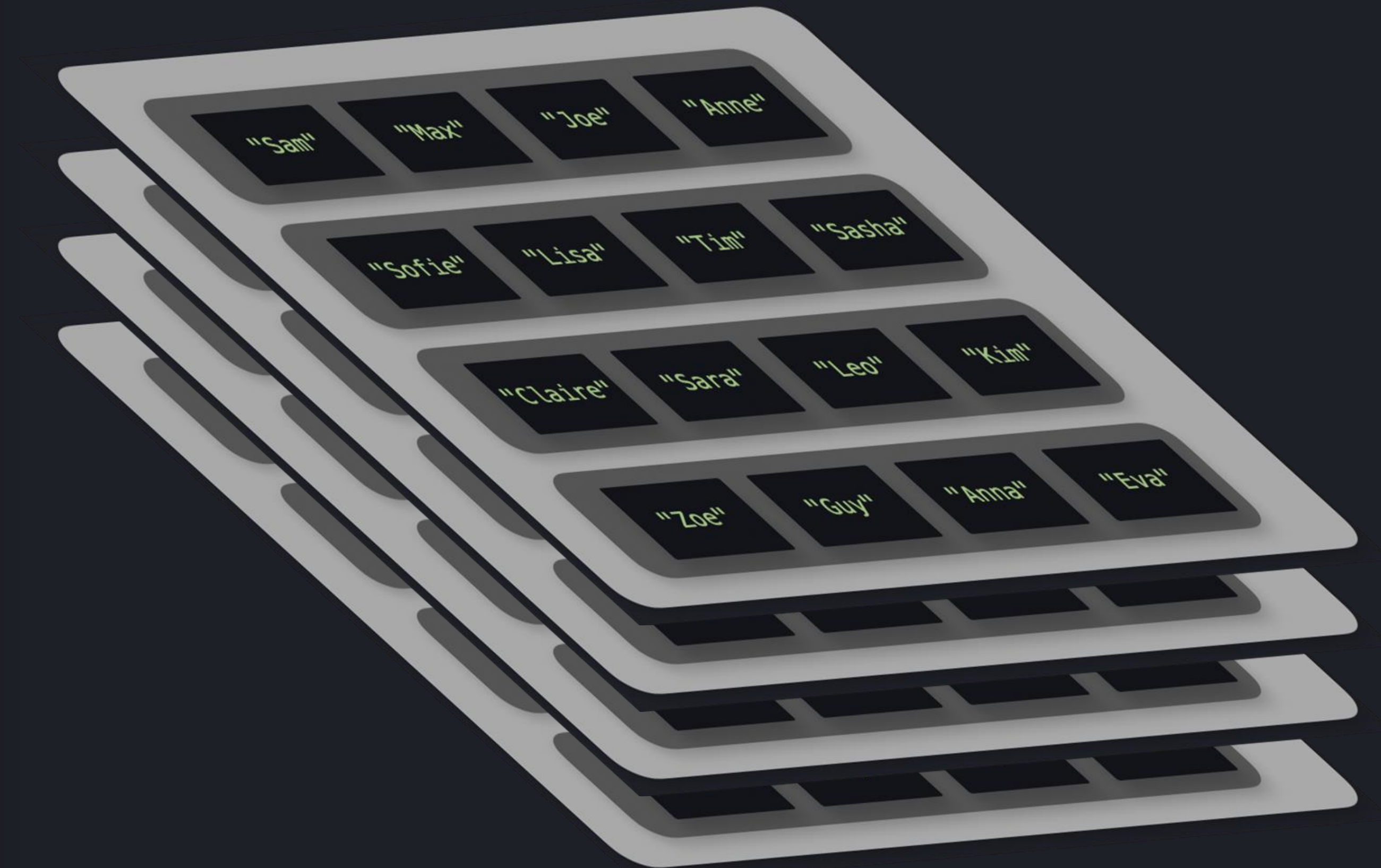




python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

```
>>> student = school[1] [2]
```

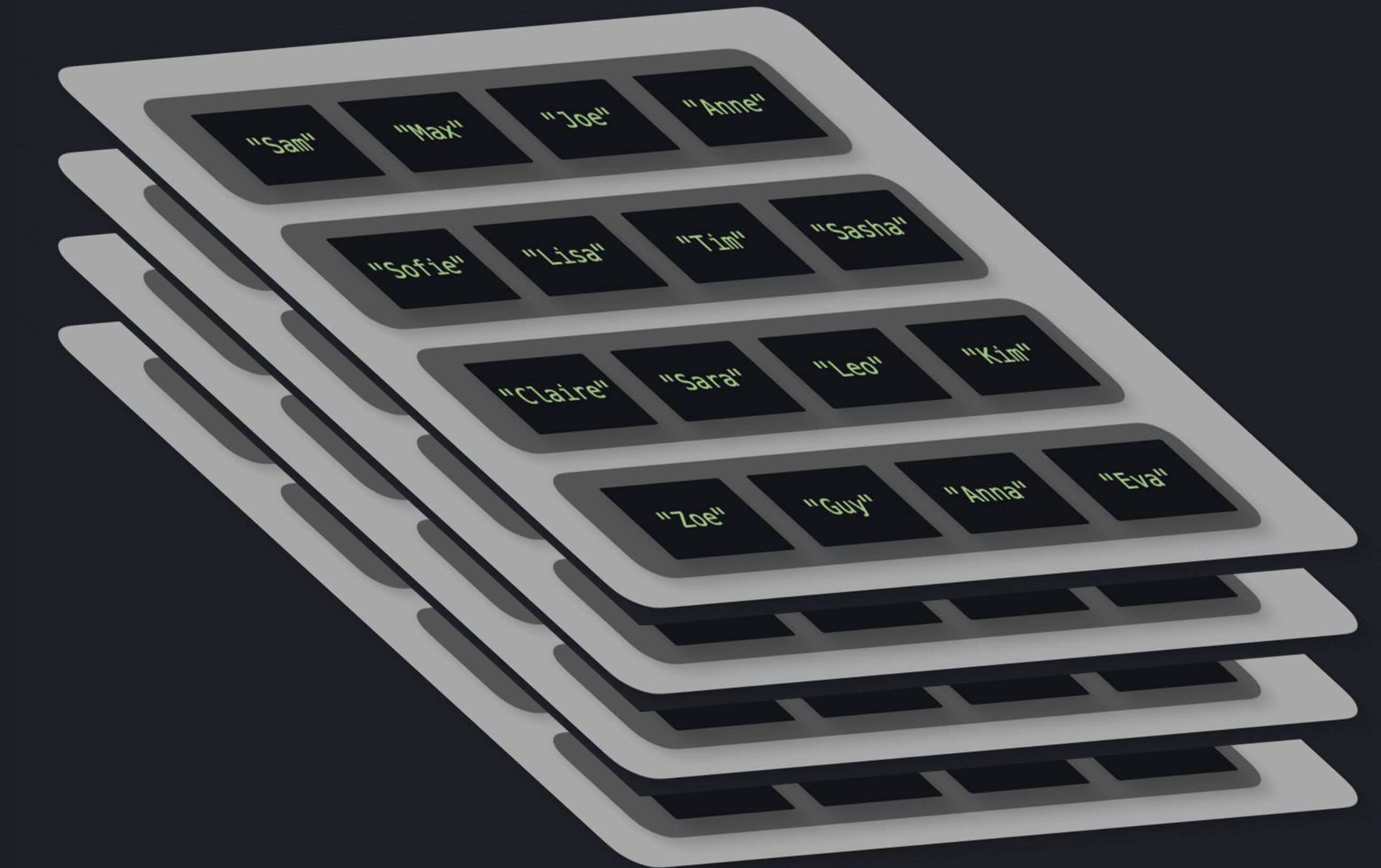




python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

```
>>> student = school[1] [2] [1]
```

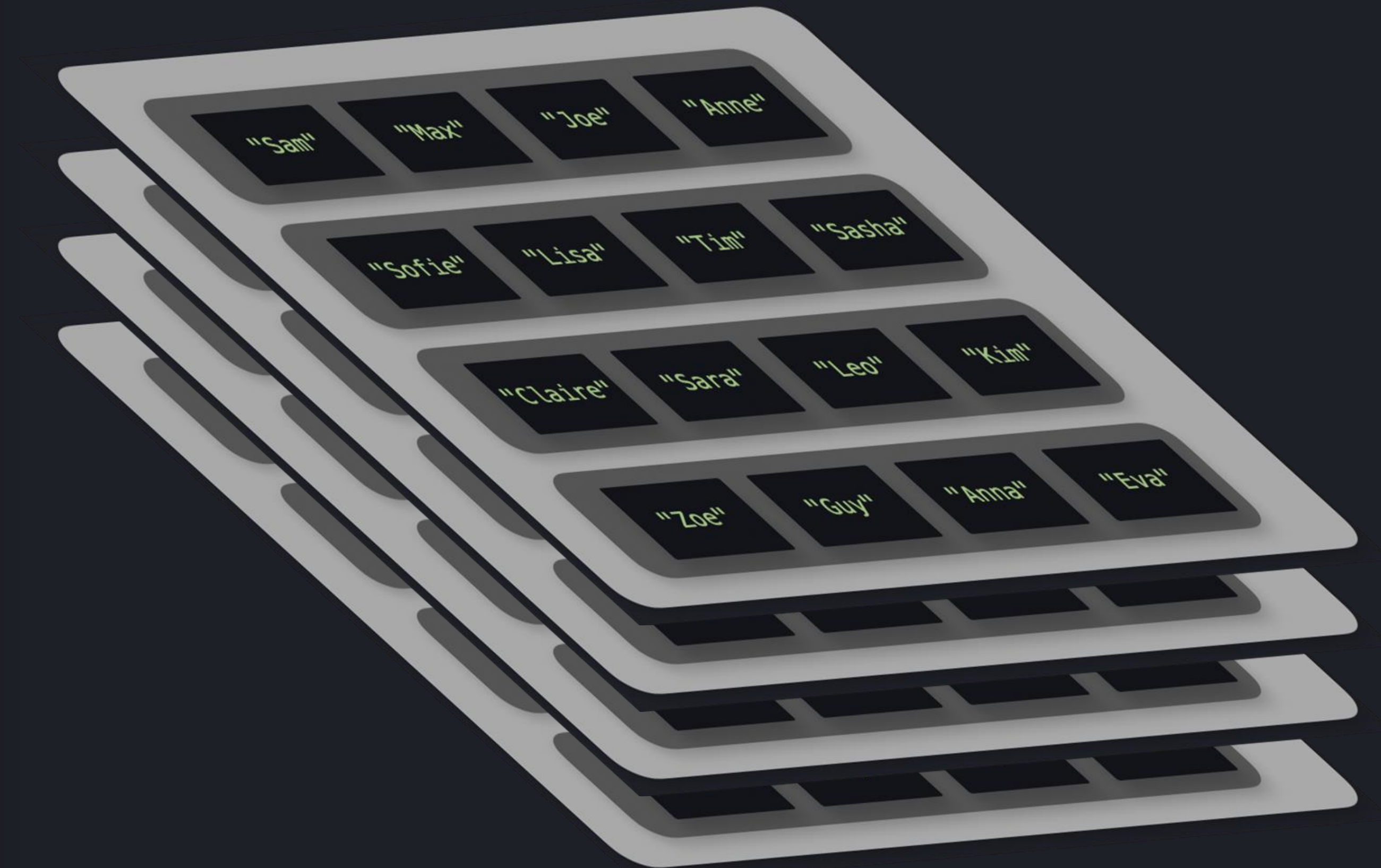




python3

```
>>> school = [  
    [  
        ["Sara", "Kim", "Anne", "Eva"],  
        ["Johan", "Collin", "Sam", "Alex"],  
        ["Luke", "Sara", "Haley", "Jennifer"],  
        ["Katy", "Mara", "Max", "Roy"],  
    ],  
    [  
        ["Anne", "Leo", "Sasha", "Tim"],  
        ["Claire", "Guy", "Eva", "Zoe"],  
        ["Lisa", "Max", "Evan", "Chloe"],  
        ["Brent", "Sam", "Sarah", "Anne"],  
    ],  
    [  
        ["Maria", "Julian", "Chris", "Tom"],  
        ["Zoe", "Anna", "Kim", "Leo"],  
        ["Vera", "Pim", "Leo", "Guy"],  
        ["Anne", "Sofie", "Max", "Joe"],  
    ],  
    [  
        ["Sam", "Max", "Joe", "Anne"],  
        ["Sofie", "Lisa", "Tim", "Sasha"],  
        ["Claire", "Sara", "Leo", "Kim"],  
        ["Zoe", "Guy", "Anna", "Eva"],  
    ],  
]
```

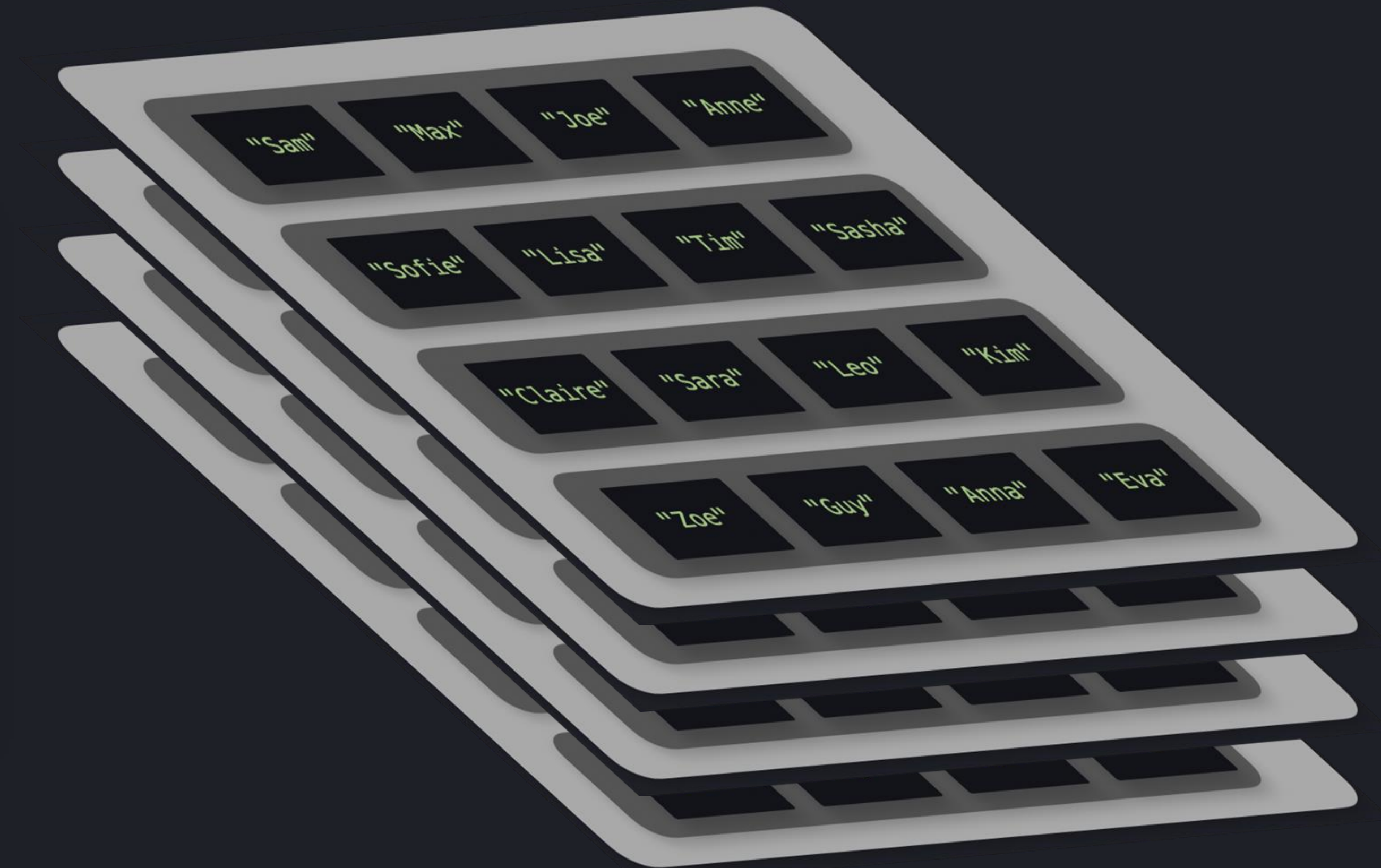
```
>>> student = school[1] [2] [1]
```





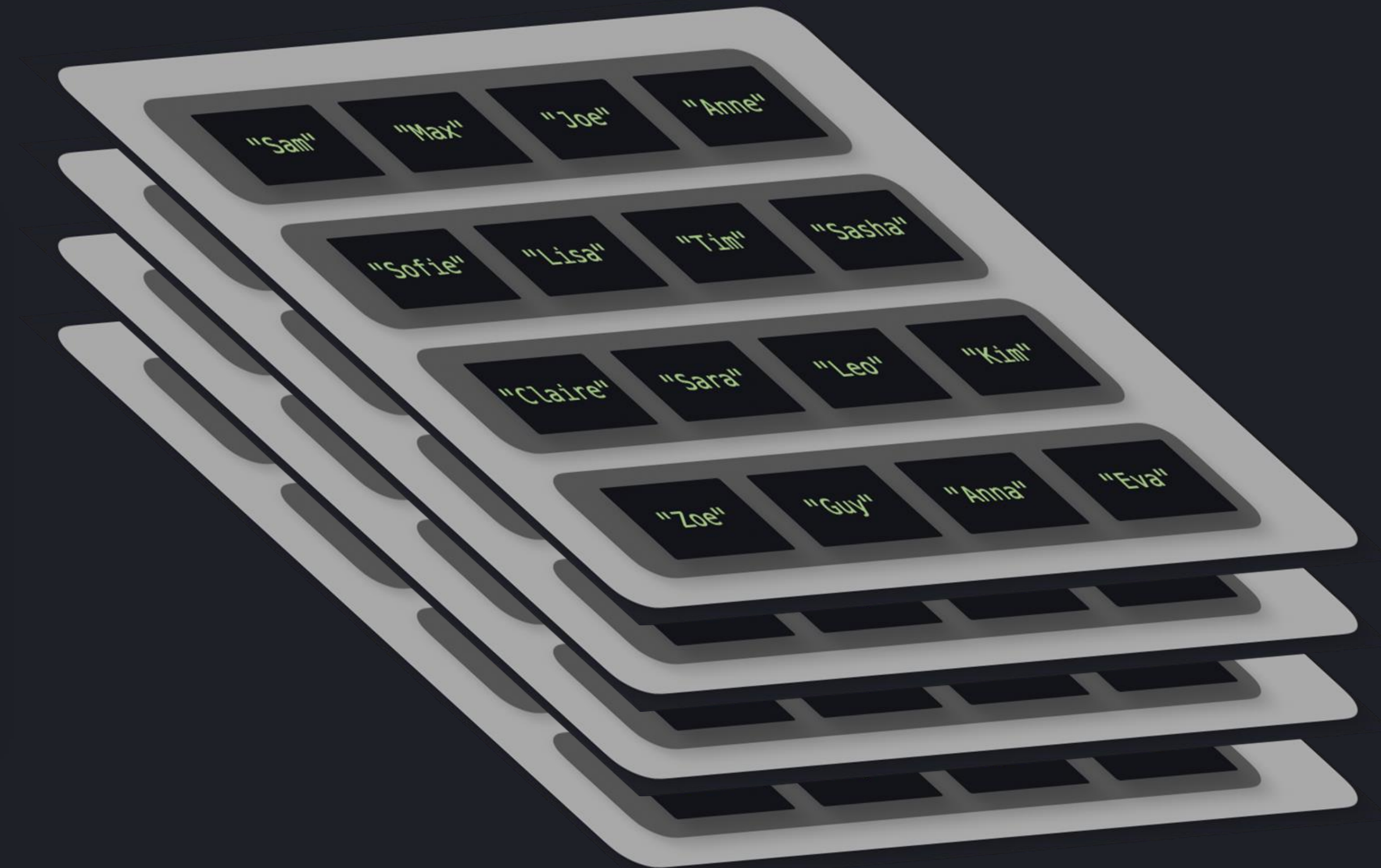
python3

```
>>> student = school[1]      [2] [1]  
>>> print(student)
```



python3

```
>>> student = school[1]      [2] [1]
>>> print(student)
"Max"
```





KodeKloud



Functions

Functions

`print()`

`len()`

`input()`

Methods

`list.append()`

`list.insert()`



python3

```
>>> input1 = int(input("Enter a number: "))
```



python3

```
>>> input1 = int(input("Enter a number: "))
```

```
>>> input2 = int(input("Enter a number: "))
```



python3

```
>>> input1 = int(input("Enter a number: "))  
>>> input2 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))
```



python3

```
>>> input1 = int(input("Enter a number: "))
>>> input2 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input4 = int(input("Enter a number: "))
```



python3

```
>>> input1 = int(input("Enter a number: "))  
>>> input2 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))  
>>> input4 = int(input("Enter a number: "))  
>>> input5 = int(input("Enter a number: "))
```




python3

```
>>> input1 = int(input("Enter a number: "))  
>>> input2 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))  
>>> input4 = int(input("Enter a number: "))  
>>> input5 = int(input("Enter a number: "))
```



python3

```
>>> input1 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input4 = int(input("Enter a number: "))
>>> input5 = int(input("Enter a number: "))
```



python3

```
>>> def input_number():
        return int(input("Enter a number: "))
```

python3

```
>>> input1 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input4 = int(input("Enter a number: "))
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

python3

```
>>>
```

python3

```
>>> input1 = int(input("Enter a number: "))
```

```
>>> input3 = int(input("Enter a number: "))
```

```
>>> input3 = int(input("Enter a number: "))
```

```
>>> input4 = int(input("Enter a number: "))
```

```
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

python3

```
>>>
```

python3

```
>>> input1 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input4 = int(input("Enter a number: "))
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
    return int(input("Enter a number: "))
```

python3

```
>>>
```

python3

```
>>> input1 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input4 = int(input("Enter a number: "))
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

python3

```
>>>
```

python3

```
>>> input1 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input4 = int(input("Enter a number: "))
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

python3

```
>>>
```

python3

```
>>> input1 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))  
>>> input4 = int(input("Enter a number: "))  
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

python3

```
>>>
```


python3

```
>>> input1 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))  
>>> input4 = int(input("Enter a number: "))  
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

python3

```
>>>
```

python3

```
>>> input1 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input3 = int(input("Enter a number: "))
>>> input4 = int(input("Enter a number: "))
>>> input5 = int(input("Enter a number: "))
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

python3

```
>>>
```



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))
```



python3

```
>>> input1 = int(input("Enter a number: "))  
>>> input2 = int(input("Enter a number: "))  
>>> input3 = int(input("Enter a number: "))  
>>> input4 = int(input("Enter a number: "))  
>>> input5 = int(input("Enter a number: "))
```



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))  
  
>>> input1 = input_number()  
>>> input2 = input_number()  
>>> input3 = input_number()  
>>> input4 = input_number()  
>>> input5 = input_number()
```



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))  
  
>>> input1 = input_number()  
>>> input2 = input_number()  
>>> input3 = input_number()  
>>> input4 = input_number()  
>>> input5 = input_number()
```



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))  
  
>>> input1 = input_number()  
>>> input2 = input_number()  
>>> input3 = input_number()  
>>> input4 = input_number()  
>>> input5 = input_number()  
>>> input1
```



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))
```

```
>>> input1 = input_number()
```

```
>>> input2 = input_number()
```

```
>>> input3 = input_number()
```

```
>>> input4 = input_number()
```

```
>>> input5 = input_number()
```

```
>>> input1
```

Enter a number: 104



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))  
  
>>> input1 = input_number()  
>>> input2 = input_number()  
>>> input3 = input_number()  
>>> input4 = input_number()  
>>> input5 = input_number()  
>>> input1  
Enter a number: 104  
>>> print(input1)
```




python3

```
>>> def input_number():  
    return int(input("Enter a number: "))  
  
>>> input1 = input_number()  
>>> input2 = input_number()  
>>> input3 = input_number()  
>>> input4 = input_number()  
>>> input5 = input_number()  
>>> input1  
Enter a number: 104  
>>> print(input1)  
104
```



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))  
  
>>> input1 = input_number()  
>>> input2 = input_number()  
>>> input3 = input_number()  
>>> input4 = input_number()  
>>> input5 = input_number()  
>>> input1  
Enter a number: 104  
>>> print(input1)  
104
```



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))  
  
>>> input1 = input_number()  
>>> input2 = input_number()  
>>> input3 = input_number()  
>>> input4 = input_number()  
>>> input5 = input_number()  
>>> input1  
Enter a number: 104  
>>> print(input1)  
104  
>>> input2
```



python3

```
>>> def input_number():  
        return int(input("Enter a number: "))
```

```
>>> input1 = input_number()
```

```
>>> input2 = input_number()
```

```
>>> input3 = input_number()
```

```
>>> input4 = input_number()
```

```
>>> input5 = input_number()
```

```
>>> input1
```

Enter a number: 104

```
>>> print(input1)
```

104

```
>>> input2
```

Enter a number: 34



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))
```

```
>>> input1 = input_number()
```

```
>>> input2 = input_number()
```

```
>>> input3 = input_number()
```

```
>>> input4 = input_number()
```

```
>>> input5 = input_number()
```

```
>>> input1
```

Enter a number: 104

```
>>> print(input1)
```

104

```
>>> input2
```

Enter a number: 34

```
>>> print(input2)
```



python3

```
>>> def input_number():  
        return int(input("Enter a number: "))
```

```
>>> input1 = input_number()
```

```
>>> input2 = input_number()
```

```
>>> input3 = input_number()
```

```
>>> input4 = input_number()
```

```
>>> input5 = input_number()
```

```
>>> input1
```

Enter a number: 104

```
>>> print(input1)
```

104

```
>>> input2
```

Enter a number: 34

```
>>> print(input2)
```

34



python3

```
>>> def input_number():  
    return int(input("Enter a number: "))
```

```
>>> input1 = input_number()
```

```
>>> input2 = input_number()
```

```
>>> input3 = input_number()
```

```
>>> input4 = input_number()
```

```
>>> input5 = input_number()
```

```
>>> input1
```

Enter a number: 104

```
>>> print(input1)
```

104

```
>>> input2
```

Enter a number: 34

```
>>> print(input2)
```

34



python3

```
>>> input1 = input_number()

def input_number():
    return int(input("Enter a number: "))
```




python3

```
>>> input1 = input_number()
```

```
def input_number():
```

```
    return int(input("Enter a number: "))
```

```
NameError: name 'input_number' is not defined
```



KodeKloud



Function - Arguments

python3

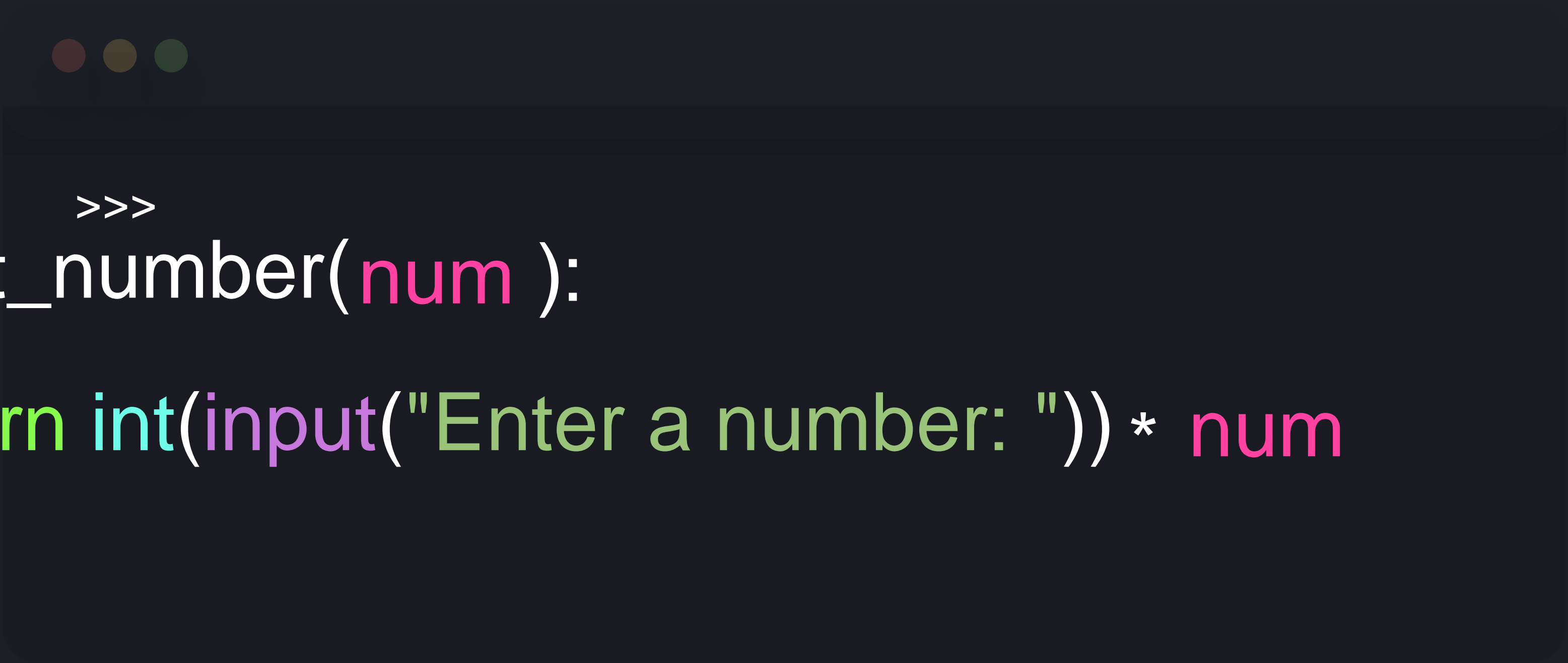
```
>>>  
def input_number(  
    return int(input("Enter a number: "))
```

python3

>>>

```
def input_number( num):
```

```
    return int(input("Enter a number: ")) * num
```



```
>>>  
def input_number(num ):  
    return int(input("Enter a number: ")) * num
```



python3

```
>>> def input_number( num ):
    return int(input("Enter a number: ")) * num
>>> input1 = input_number(10)
```



python3

```
>>> def input_number( num ):
    return int(input("Enter a number: ")) * num

>>> input1 = input_number(10)
>>> input1
```




python3

```
>>> def input_number( num ):
        return int(input("Enter a number: ")) * num
>>> input1 = input_number(10)
>>> input1
Enter a number: 12
```



python3

```
>>> def input_number( num ):
        return int(input("Enter a number: ")) * num

>>> input1 = input_number(10)
>>> input1
Enter a number: 12
>>> print(input1)
```



python3

```
>>> def input_number( num ):
        return int(input("Enter a number: ")) * num

>>> input1 = input_number(10)
>>> input1
Enter a number: 12
>>> print(input1)
120
```



python3

```
>>> def input_number(num1, num2):  
    return int(input("Enter a number: ")) * num1 - num2
```



python3

```
>>> def input_number(num1, num2):  
    return int(input("Enter a number: ")) * num1 - num2  
  
>>> input1 = input_number(10, 20)
```

python3

```
>>>def input_number(num1, num2):  
    return int(input("Enter a number: ")) * num1 - num2  
  
input1 = input_number( 10, 20)
```

python3

```
def input_number(num1, num2):  
    return int(input("Enter a number: ")) * num1 - num2  
  
input1 = input_number(10, 20)
```

```
graph TD; num1_def[num1] -- cyan --> 10[10]; num2_def[num2] -- orange --> 20[20]; 10 -- cyan --> input_number; 20 -- orange --> input_number; input_number --> input1[input1];
```

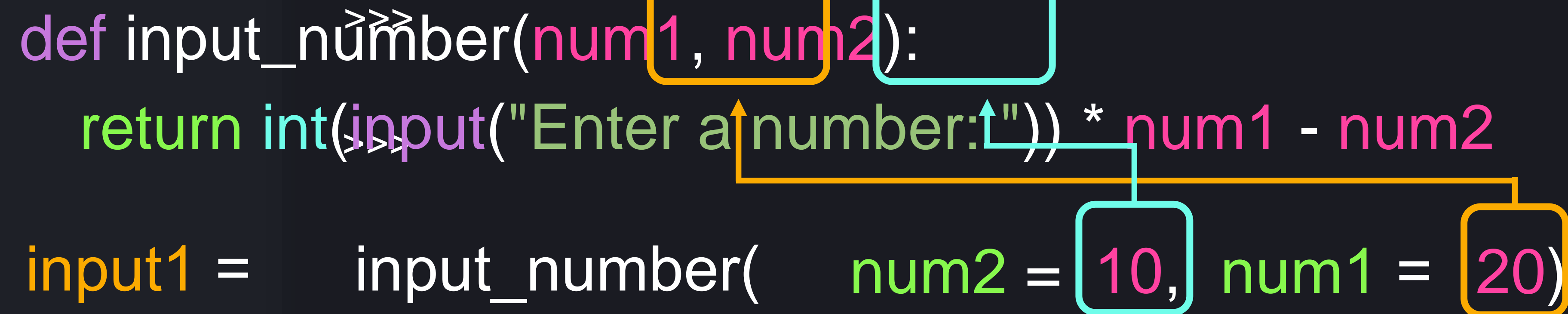
python3

```
>>>
def input_number(num1, num2):
    return int(input("Enter a number: ")) * num1 - num2

input1 = input_number(num2 = 10, num1 = 20)
```


python3

```
def input_number(num1, num2):  
    return int(input("Enter a number:")) * num1 - num2  
  
input1 = input_number(num2 = 10, num1 = 20)
```



python3

```
def input_number(num1, num2):  
    return int(input("Enter a number: ")) * num1 - num2  
  
input1 = input_number(10, num1 = 20)
```



python3

```
>>> def input_number(num1, num2):  
    return int(input("Enter a number: ")) * num1 - num2  
  
>>> input1 = input_number(10, num1 = 20)
```

TypeError: input_number() got multiple values
for argument 'num1'



python3

```
>>> def input_number( num ):
      return int(input("Enter a number: ")) * num
```



python3

```
>>> def input_number( num = 10 ):
      return int(input("Enter a number: ")) * num
```



python3

```
>>> def input_number( num = 10 ):
        return int(input("Enter a number: ")) * num
>>> input_number()
```



python3

```
>>> def input_number( num = 10 ):
        return int(input("Enter a number: ")) * num
```

```
>>> input_number()
```

```
Enter a number: 12
```



python3

```
>>> def input_number( num = 10 ):
        return int(input("Enter a number: ")) * num
```

```
>>> input_number()
```

```
Enter a number: 12
```

```
120
```




python3

```
>>> def input_number( num = 10 ):
        return int(input("Enter a number: ")) * num
```

```
>>> input_number()
```

Enter a number: 12

120



python3

```
>>> def input_number(    num  = 10 ):
        return int(input("Enter a number: ")) * num
>>> input_number(5)
```



python3

```
>>> def input_number( num = 10 ):
        return int(input("Enter a number: ")) * num

>>> input_number(5)
Enter a number: 12
```



python3

```
>>> def input_number( num = 10 ):
        return int(input("Enter a number: ")) * num
```

```
>>> input_number(5)
```

```
Enter a number: 12
```

```
60
```



python3

```
>>> def input_number( num = 10 ):
        return int(input("Enter a number: ")) * num
```

```
>>> input_number(5)
```

Enter a number: 12

60



KodeKloud



Function - Return



python3

```
>>> def input_number( num = 10 ):
      return int(input("Enter a number: ")) * num
```




python3

```
>>> def print_sum(    num1, num2 ):
      sum = num1 + num2
      print("The sum is: ", str(sum))
```



python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2
        print("The sum is: ", str(sum))
>>> print_sum(10, 20)
```



python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2
        print("The sum is: ", str(sum))
>>> print_sum(10, 20)
The sum is: 30
```



python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2
        print("The sum is: ", str(sum))
>>> print_sum(10, 20)
The sum is: 30
```



python3

```
>>> def print_sum(    num1, num2 ):
    sum = num1 + num2
    return
    print("The sum is: ", str(sum))
```



python3

```
>>> def print_sum(    num1, num2 ):
    sum = num1 + num2
    return
    print("The sum is: ", str(sum))
```



python3

```
>>> def print_sum(    num1, num2 ):
    sum = num1 + num2

    if(sum == 0):
        return

    print("The sum is: ", str(sum))
```



python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2

        if(sum == 0):
            return

        print("The sum is: ", str(sum))

>>> print_sum(4, 2)
```




python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2

        if(sum == 0):
            return

        print("The sum is: ", str(sum))

>>> print_sum(4, 2)
The sum is: 6
```



python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2

        if(sum == 0):
            return

        print("The sum is: ", str(sum))

>>> print_sum(4, 2)
The sum is: 6

>>> print_sum(-1, 1)
```



python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2

        if(sum == 0):
            return

        print("The sum is: ", str(sum))

>>> print_sum(4, 2)
The sum is: 6

>>> print_sum(-1, 1)

>>>
```

python3

```
>>> def print_sum(    num1, num2 ):
    sum = num1 + num2

    if(sum == 0):
        return

    print("The sum is: ", str(sum))

>>> print_sum(4, 2)
The sum is: 6

>>> print_sum(-1, 1)

>>>
```



python3

```
>>> def print_sum(    num1, num2 ):
        sum = num1 + num2

        if(sum == 0):
            return

        print("The sum is: ", str(sum))

>>> print_sum(4, 2)
The sum is: 6

>>> print_sum(-1, 1)

>>>
```



python3

```
>>> def is_even( num ):
      if(num % 2 == 0):
          return True
```



python3

```
>>> def is_even( num ):
        if(num % 2 == 0):
            return True

>>> print(is_even(6))
```



python3

```
>>> def is_even( num ):
        if(num % 2 == 0):
            return True
```

```
>>> print(is_even(6))
True
```




python3

```
>>> def is_even( num ):
        if(num % 2 == 0):
            return True
```

```
>>> print(is_even(6))
True
```

```
>>> print(is_even(7))
```



python3

```
>>> def is_even( num ):
        if(num % 2 == 0):
            return True
```

```
>>> print(is_even(6))
True
```

```
>>> print(is_even(7))
None
```



KodeKloud



Function - List as Argument



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values
```



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values
```



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
    for item in list:  
        multiplied_values.append(item * 2)  
    return multiplied_values
```



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values
```




python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values
```



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values
```



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values  
  
>>> print(multiply_values([1, 2, 3]))
```

python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values  
  
>>> print(multiply_values([1, 2, 3]))  
[2, 4, 6]
```

python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values  
  
>>> print(multiply_values([1, 2, 3]))  
[2, 4, 6]  
>>> print(multiply_values([-4, -8, -10]))
```

python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values
```

```
>>> print(multiply_values([1, 2, 3]))  
[2, 4, 6]
```

```
>>> print(multiply_values([-4, -8, -10]))  
[-8, -16, -20]
```



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values  
  
>>> print(multiply_values([1, 2, 3]))  
[2, 4, 6]  
>>> print(multiply_values([-4, -8, -10]))  
[-8, -16, -20]  
>>> print(multiply_values(1))
```



python3

```
>>> def multiply_values(list):  
    multiplied_values = []  
  
    for item in list:  
        multiplied_values.append(item * 2)  
  
    return multiplied_values
```

```
>>> print(multiply_values([1, 2, 3]))  
[2, 4, 6]
```

```
>>> print(multiply_values([-4, -8, -10]))  
[-8, -16, -20]
```

```
>>> print(multiply_values(1))
```

TypeError: 'int' object is not iterable



KodeKloud



Scopes



python3

```
>>> def input_number():  
    result = int(input("Enter a number: ")) * 100  
    return result
```



python3

```
>>> def input_number():  
    result = int(input("Enter a number: ")) * 100  
    return result
```



python3

```
>>> def input_number():  
    result = int(input("Enter a number: ")) * 100  
    return result  
  
>>> print(result)
```



python3

```
>>> def input_number():  
    result = int(input("Enter a number: ")) * 100  
    return result  
  
>>> print(result)
```

NameError: name 'result' is not defined



python3

```
>>> num = 100
```

```
>>> def input_number():  
    result = int(input("Enter a number: "))  
    return result
```

nu
*
m



python3

```
>>> num = 100
```

```
>>> def input_number():  
    num = 50  
    result = int(input("Enter a number: "))  
    return result
```

nu
*
m



python3

```
>>> num = 100
```

```
>>> def input_number():  
    num = 50  
    result = int(input("Enter a number: "))  
    return result
```

nu
*
m



python3

```
>>> num = 100
```

```
>>> def input_number():  
    result = int(input("Enter a number: "))  
    return result
```

nu
*
m



python3

```
>>> num = 100
```

```
>>> def input_number():
```

```
    own_num = 50
```

```
    result = int(input("Enter a number: "))
```

```
    return result
```

```
>>> print(own_num)
```

own_nu
*
m



python3

```
>>> num = 100
```

```
>>> def input_number():
```

```
    own_num = 50
```

```
    result = int(input("Enter a number: "))
```

```
    return result
```

```
>>> print(own_num)
```

```
NameError: name 'own_num' is not defined
```

own_nu
*
m



python3

```
>>> num = 100
```

```
>>> def input_number():  
    global  
    own_numm = 50  
    result = int(input("Enter a number: "))  
    return result
```

```
    * own_nu  
    m
```



python3

```
>>> num = 100
>>> def input_number():
    global own_num
    own_num = 50
    result = int(input("Enter a number: ")) * own_num
    return result
>>> print(own_num)
```



python3

```
>>> num = 100
```

```
>>> def input_number():  
    global own_num  
    own_num = 50  
    result = int(input("Enter a number: ")) * own_num  
    return result
```

```
>>> print(own_num)
```

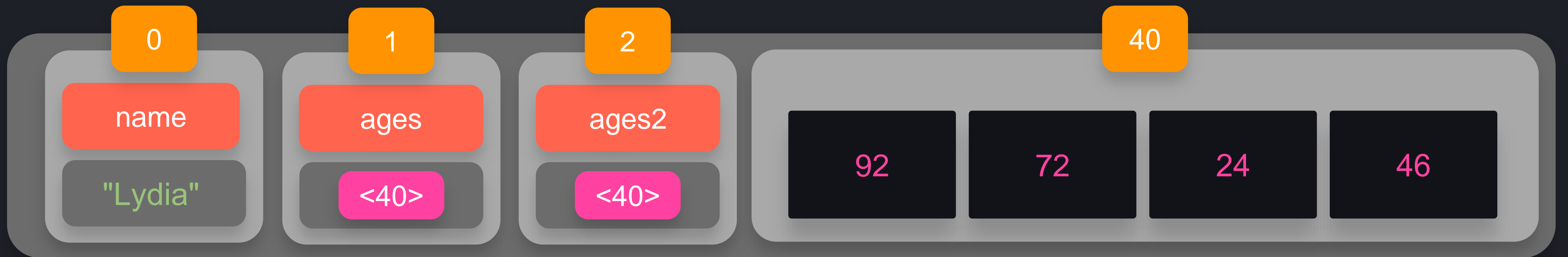
```
50
```



KodeKloud



Function - Arguments Explained



```
python3

>>> name = "Lydia"
>>> ages = [56, 72, 24, 46]
>>> ages2 = ages
>>> ages[0] = 92
>>> print(ages2[0])
92
```



python3

```
>>> age = 22
>>> def multiply(num):
    num *= 2
    print("In multiply: ", str(num))
```



python3

```
>>> age = 22
>>> def multiply(num):
    num *= 2
    print("In multiply: ", str(num))
>>> multiply(age)
```



python3

```
>>> age = 22
>>> def multiply(num):
    num *= 2
    print("In multiply: ", str(num))
>>> multiply(age)
In multiply: 44
```



python3

```
>>> age = 22
>>> def multiply(num):
    num *= 2
    print("In multiply: ", str(num))
>>> multiply(age)
In multiply: 44
>>> print(age)
```



python3

```
>>> age = 22
>>> def multiply(num):
    num *= 2
    print("In multiply: ", str(num))
>>> multiply(age)
In multiply: 44
>>> print(age)
22
```

0

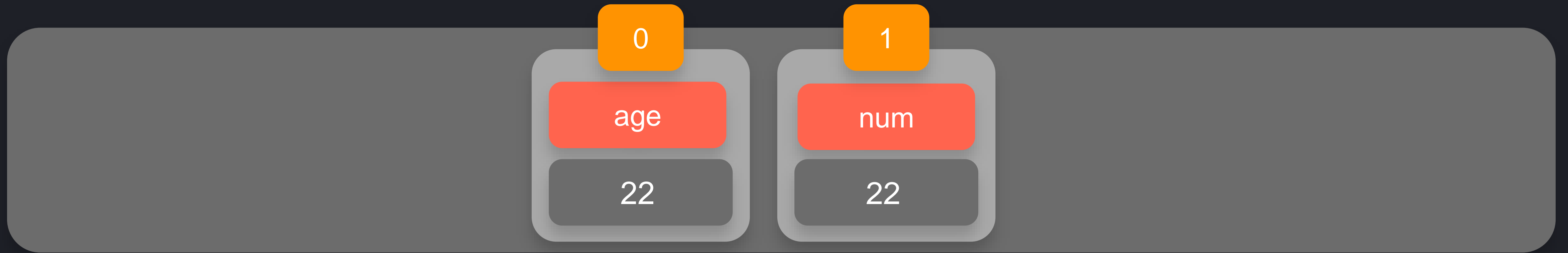
age

22



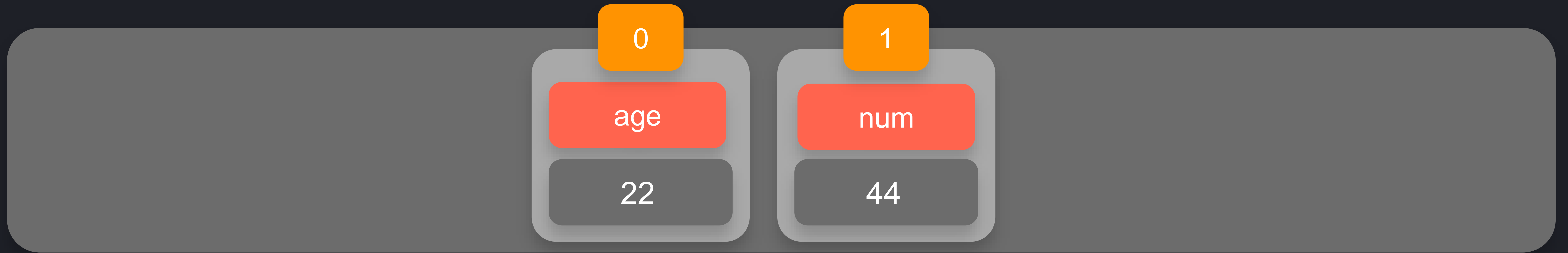
python3

```
>>> age = 22
>>> def multiply(num):
    num *= 2
    print("In multiply: ", str(num))
>>> multiply(age)
```

python3

```
>>> age = 22
>>> def multiply(nu
               nu
               m
               ):
    *= 2
    m
    print("In multiply: ", str(num))
>>> multiply(age)
```



python3

```
>>> age = 22
>>> def multiply(nu
               nu
               m
               ):
    *= 2
    m
    print("In multiply: ", str(num))
>>> multiply(age)
```

0

age

22

1

num

44



python3

```
>>> age = 22
>>> def multiply(nu
               nu
               m
               ):
    *= 2
    m
    print("In multiply: ", str(num))
>>> multiply(age)
In multiply: 44
```

0

age

22

1

num

44



python3

```
>>> age = 22
>>> def multiply(num):
    num *= 2
    print("In multiply: ", str(num))
>>> multiply(age)
In multiply: 44
>>> print(age)
22
```

0

nums

<40>

40

1

2

3

python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
```

0

nums

<40>

40

1

2

3

python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
```

0

nums

<40>

40

1

2

3

python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
```

0

nums

<40>

40

1

2

3

python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
>>> change_first_item(nums)
```




python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
>>> change_first_item(nums)
```



python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
>>> change_first_item(nums)
```



python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
>>> change_first_item(nums)
>>> print(nums)
```



python3

```
>>> nums = [1, 2, 3]
>>> def change_first_item(list):
    list[0] = 9
>>> change_first_item(nums)
>>> print(nums)
[9, 2, 3]
```



KodeKloud



Tuples

[1, 2, 3]

```
[1, 2, 3]
```

```
del myList[1]
```

```
myList.append(4)
```




python3

```
>>> tuple1 = (1, 2, 3)
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> print(tuple1)
```



python3

```
>>> tuple1 = (1, 2, 3)
>>> print(tuple1)
(1, 2, 3)
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> print(tuple1)
```

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

```
>>> tuple2 = 1, 2, 3
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> print(tuple1)  
(1, 2, 3)
```

```
>>> tuple2 = 1, 2, 3
```

```
>>> print(tuple2)
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> print(tuple1)
```

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

```
>>> tuple2 = 1, 2, 3
```

```
>>> print(tuple2)
```

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



python3

```
>>> tuple1 = (1, 2, 3)
```



python3

```
>>> tuple1 = (1, 2, 3)
>>> for item in tuple1:
    print(item)
```




python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> for item in tuple1:  
    print(item)
```

1

2

3



python3

```
>>> tuple1 = (1, 2, 3)
>>> for item in tuple1:
    print(item)
1
2
3
>>> print(tuple1[0:1])
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> for item in tuple1:  
    print(item)
```

1

2

3

```
>>> print(tuple1[0:1])
```

(0, 1)



python3

```
>>> tuple1 = (1, 2, 3)
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> tuple1.append(4)
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> tuple1.append(4)
```

```
AttributeError: 'tuple' object has no attribute 'append'
```



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> tuple1.append(4)
```

AttributeError: 'tuple' object has no attribute 'append'

```
>>> tuple1[4] = 9
```

TypeError: 'tuple' object does not support item assignment



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> tuple1.append(4)
```

AttributeError: 'tuple' object has no attribute 'append'

```
>>> tuple1[4] = 9
```

TypeError: 'tuple' object does not support item assignment

```
>>> del tuple1[1]
```

TypeError: 'tuple' object doesn't support item deletion



python3

```
>>> tuple1 = (1, 2, 3)
```

```
>>> tuple1.append(4)
```

AttributeError: 'tuple' object has no attribute 'append'

```
>>> tuple1[4] = 9
```

TypeError: 'tuple' object does not support item assignment

```
>>> del tuple1[1]
```

TypeError: 'tuple' object doesn't support item deletion



python3

```
>>> age = 22
```

```
>>> tuple1 = (1, "Lydia", age, (1, 2))
```



python3

```
>>> tuple1 = (1,)
```

```
>>> tuple2 = 1,
```



KodeKloud



Dictionaries

| Name | Username |
|-------|-------------|
| lydia | lydiahallie |
| sarah | sarah123 |
| max | max |
| joe | joejoe |

python3

```
>>>
{"lydia": "lydiahallie",
 "sarah": "sarah123",
 "max": "max_",
 "joe": "joejoe",
 }
```

| Name | Username |
|-------|-------------|
| lydia | lydiahallie |
| sarah | sarah123 |
| max | max_ |
| joe | joejoe |



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```




python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> print(usernames["sarah"])
```

```
"sarah123"
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> print(usernames["anotherone"])
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> print(usernames["anotherone"])
```

```
KeyError: 'anotherone'
```

Methods

dictionary.keys()

dictionary.values()

dictionary.items()



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> print(usernames.keys())
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> print(usernames.keys())
```

```
dict_keys(['lydia', 'sarah', 'max', 'joe'])
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> print(usernames.keys())
```

```
dict_keys(['lydia', 'sarah', 'max', 'joe'])
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
  
>>> for key in usernames.keys():  
    print(key + " - " + usernames[key])
```




python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
  
>>> for key in usernames.keys():  
    print(key + " - " + usernames[key])  
  
lydia - lydiahallie  
sarah - sarah123  
max - max_  
joe - joejoe
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
>>> print(usernames.values())
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
  
>>> print(usernames.values())  
  
dict_values(['lydiahallie', 'sarah123', 'max_', 'joejoe'])
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
>>> print(usernames.items())
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
  
>>> print(usernames.items())  
  
dict_items([  
    ('lydia', 'lydiahallie'),  
    ('sarah', 'sarah123'),  
    ('max', 'max_'),  
    ('joe', 'joejoe')  
])
```

Methods

```
dictionary.keys()
```

```
dictionary.values()
```

```
dictionary.items()
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> usernames["max"] = "max123"
```




python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> usernames["max"] = "max123"
```

```
>>> print(usernames["max"])
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> usernames["max"] = "max123"
```

```
>>> print(usernames["max"])  
"max123"
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
  
>>> usernames.update({ "chloe": "chloe123" })
```



python3

```
>>>
    usernames = {
        "lydia": "lydiahallie",
        "sarah": "sarah123",
        "max": "max_",
        "joe": "joejoe",
    }
>>> usernames.update({ "chloe": "chloe123" })
>>> print(usernames)
```



python3

```
>>>
    usernames = {
        "lydia": "lydiahallie",
        "sarah": "sarah123",
        "max": "max_",
        "joe": "joejoe",
    }
>>> usernames.update({ "chloe": "chloe123" })
>>> print(usernames)
{
    "lydia": "lydiahallie",
    "sarah": "sarah123",
    "max": "max_",
    "joe": "joejoe",
    "chloe": "chloe123"
}
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
>>> del usernames["max"]
```




python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}
```

```
>>> del usernames["max"]
```

```
>>> print(usernames)
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}
```

```
>>> del usernames["max"]
```

```
>>> print(usernames)  
{  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "joe": "joejoe"  
}
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
>>> usernames.clear()
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
  
>>> usernames.clear()  
  
>>> print(usernames)
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}
```

```
>>> usernames.clear()
```

```
>>> print(usernames)  
{
```



python3

```
>>> usernames = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}  
  
>>> usernames.popitem()  
  
>>> print(usernames)
```



python3

```
>>> usernames = {
    "lydia": "lydiahallie",
    "sarah": "sarah123",
    "max": "max_",
    "joe": "joejoe",
}

>>> usernames.popitem()

>>> print(usernames)
{
    "lydia": "lydiahallie",
    "sarah": "sarah123",
    "max": "max_"
}
```



python3

```
>>>
```

```
    usernames = {  
        "lydia": "lydiahallie",  
        "sarah": "sarah123",  
        "max": "max_",  
        "joe": "joejoe",  
    }
```

```
>>> usernames_copy = usernames.copy()
```




python3

```
>>>
```

```
username = {  
    "lydia": "lydiahallie",  
    "sarah": "sarah123",  
    "max": "max_",  
    "joe": "joejoe",  
}
```

```
>>> username_copy = username.copy()
```

```
>>> print(username_copy)
```



python3

```
>>>
    usernames = {
        "lydia": "lydiahallie",
        "sarah": "sarah123",
        "max": "max_",
        "joe": "joejoe",
    }
>>> usernames_copy = usernames.copy()
>>> print(usernames_copy)
{
    "lydia": "lydiahallie",
    "sarah": "sarah123",
    "max": "max_",
    "joe": "joejoe"
}
```



KodeKloud