# AIBridge

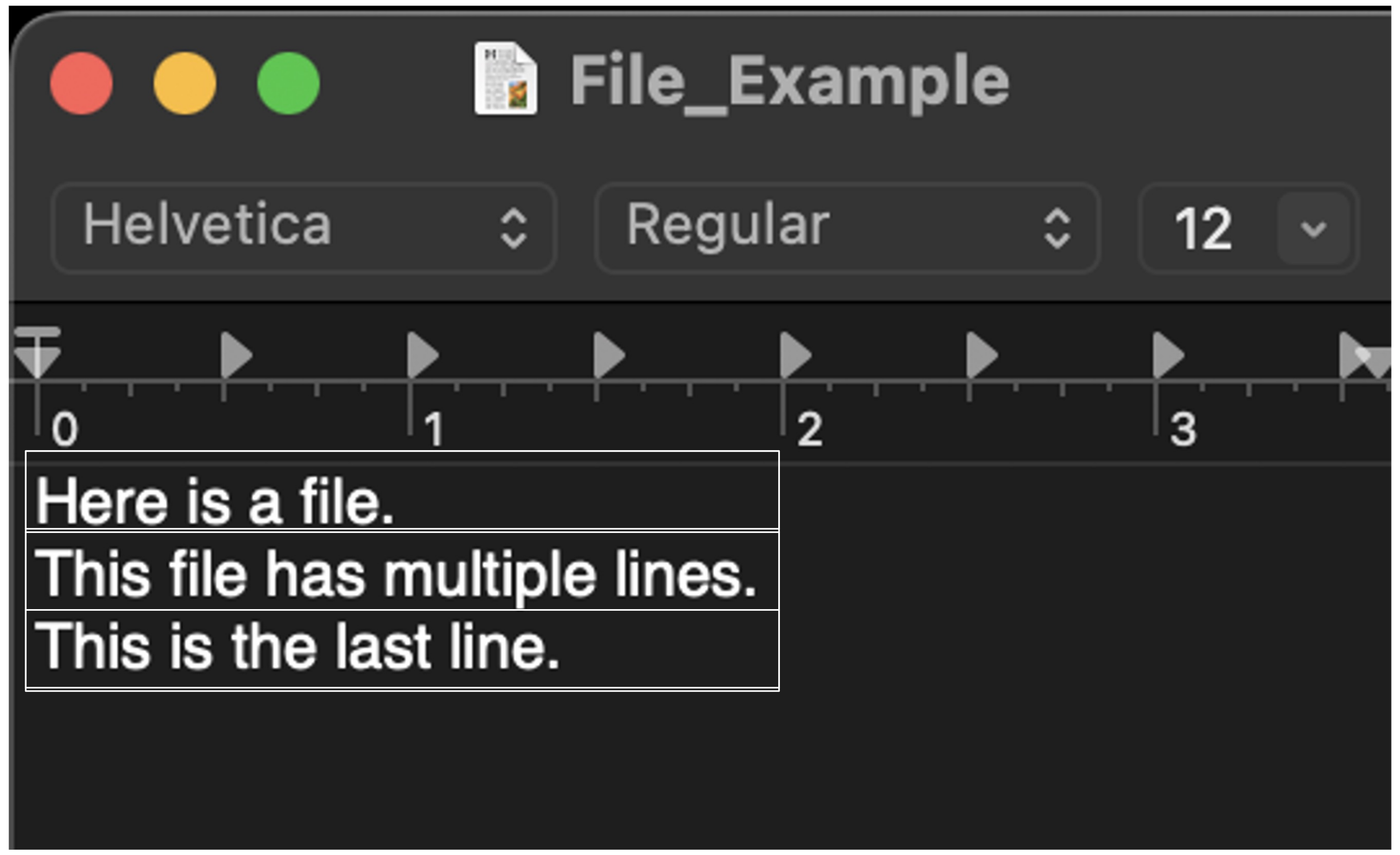
## Lecture 2

## Lecture Outline

I/O

List Manipulation

# I/O
## Standard Input



Here is a file.
This file has multiple lines.
This is the last line.

Input from console: `input('prompt')`

Open file: `file_object=open(file, mode)`

`'r'` is read and `'w'` is write for the mode
`read(), readline(), readlines()`

**Always close file:** `file_object.close()`

```
"""Here is a file.
This file has multiple lines.
This is the last line."""
```

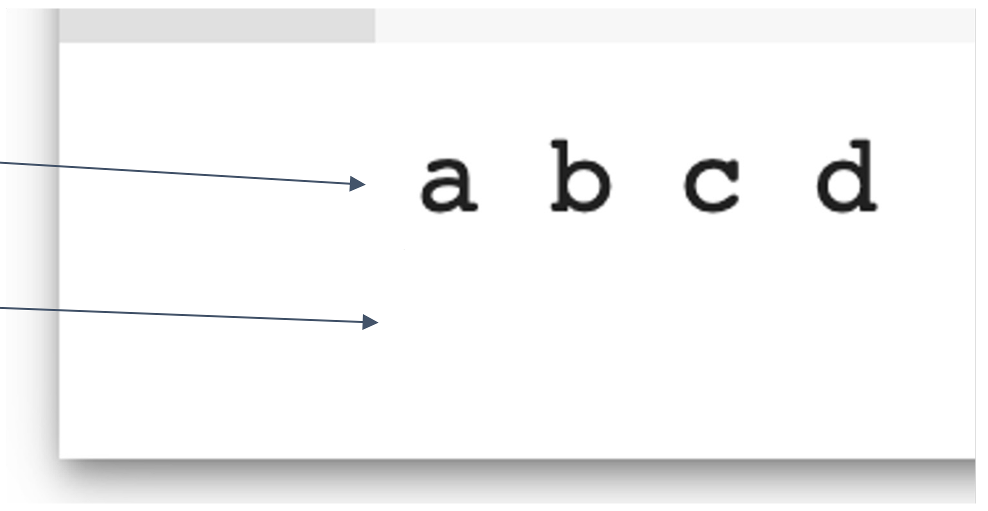
```
"Here is a file."
"This file has multiple lines."
"This is the last line."
```

```
["Here is a file.",
"This file has multiple lines.",
"This is the last line."]
```

## I/O
## Standard Output

Output to Console: `print(object1, object2, ...)`

```
print('a', 'b', 'c', 'd')
print('e', 'f', 'g')
```



```
a b c d
```

Open file: `file_object=open(file, mode)`

`write()`

**Always** close file

**Note: This removes any existing file with that name**

## Lecture Outline

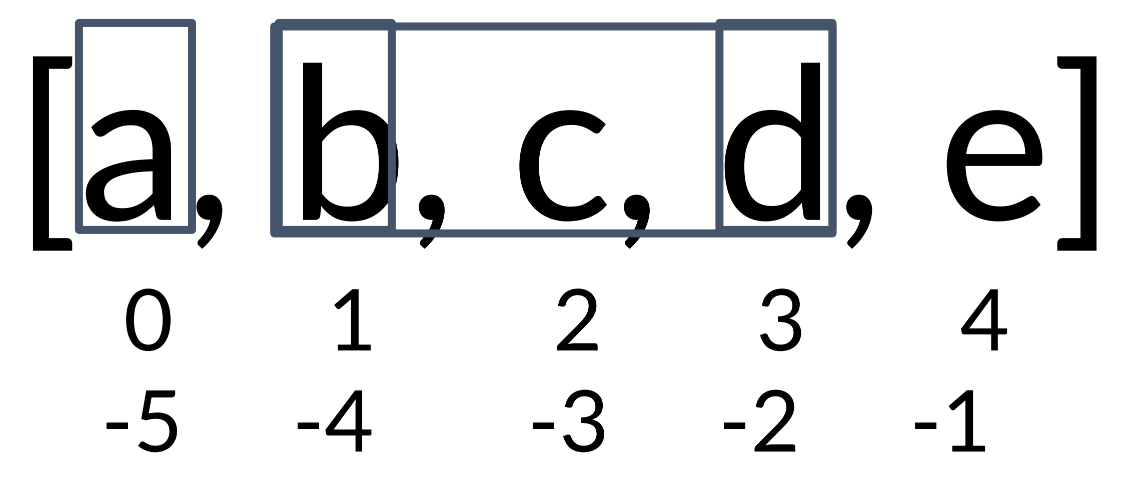
I/O

List Manipulation

## List Manipulation

Indexing
List Operations
Listcomp
String/list Interop
Multidimensional Lists

## List Manipulation
## Indexing



Single indexing

```
list_name[0]
```

```
list_name[-2]
```

List slicing

```
list_name[1:4]
```

## List Manipulation
## Indexing

### Self-Test

What does the following code output?

```
arr = [4, 5, 6, 101, 102, 103, 104, 105]


new_arr = arr[2:6]
print(new_arr)
```

```
A. [5, 6, 7, 101, 102, 103, 104, 105]
B. [6, 7, 101, 102, 103, 104, 105]
C. [6, 101, 102, 103, 104]
D. [6, 101, 102, 103]
```

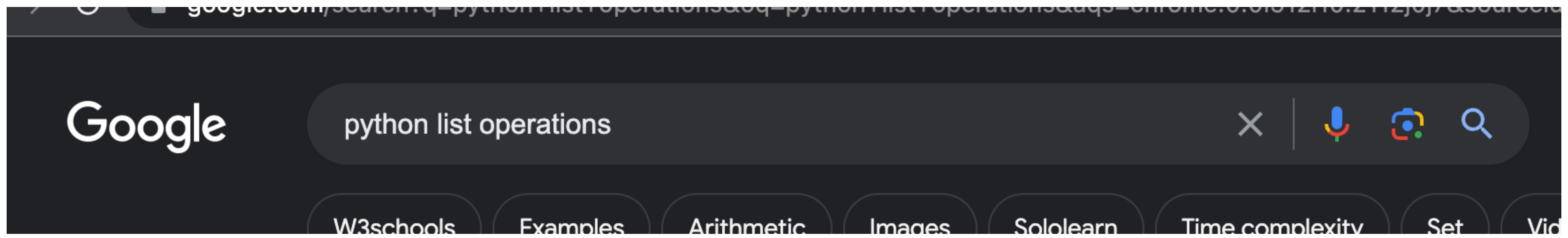
## List Manipulation

Indexing
List Operations
Listcomp
String/list Interop
Multidimensional Lists

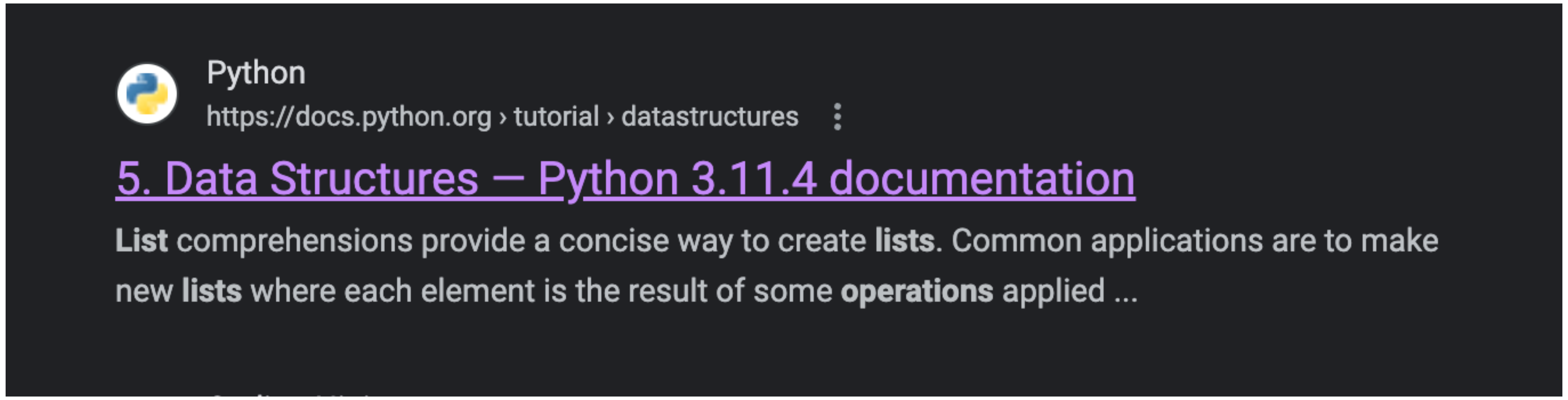
# List Manipulation
# List Operations

https://docs.python.org/3/tutorial/datastructures.html

①



②


Python
https://docs.python.org › tutorial › datastructures

5. Data Structures — Python 3.11.4 documentation

**List** comprehensions provide a concise way to create **lists**. Common applications are to make new **lists** where each element is the result of some **operations** applied ...

## List Manipulation
## List Operations

```
my_list = [3, 14, 0, -2, 5]
```

## List Manipulation
## List Operations
### append()

```
[3, 14, 0, -2, 5]
```

```
my_list.append(19)
```

## List Manipulation
## List Operations
### append()

```
[3, 14, 0, -2, 5, 19]
```

```
my_list.append(19)
my_list.append(8)
```

## List Manipulation
## List Operations
### append()

```
[3, 14, 0, -2, 5, 19, 8]
```

```
my_list.append(19)
my_list.append(8)
```

## List Manipulation
## List Operations
### remove()

```
[3, 14, 0, -2, 5, 19, 8]
```

```
my_list.remove(-2)
```

## List Manipulation
## List Operations
### remove()

```
[3, 14, 0, 5, 19, 8]
```

$\boxed{\texttt{3,}}$

```
my_list.remove(-2)
my_list.remove(19)
```

## List Manipulation
## List Operations
### remove()

```
[3, 14, 0, 5, 8]
```

```
my_list.remove(-2)
my_list.remove(19)
```

## List Manipulation
## List Operations
### insert()

```
[3, 14, 0, 5, 8]
```

```
my_list.insert(3, 14)
```

## List Manipulation
## List Operations
### insert()

```
[3, 14, 0, 14, 5, 8]
```

```
my_list.insert(3, 14)
my_list.insert(3, 1)
```

## List Manipulation
### List Operations
### insert()

```
[3, 14, 0, 1, 14, 5, 8]
```

```
my_list.insert(3, 14)
my_list.insert(3, 1)
```

## List Manipulation
## List Operations
### pop()

```
[3, 14, 0, 1, 14, 5, 8]
```

```
my_list.pop(3)
```

## List Manipulation
## List Operations
### pop()

```
[3, 14, 0, 14, 5, 8]
```

$$\texttt{my\_list.pop(3)} \longrightarrow 1$$

```
my_list.pop(3)
```

## List Manipulation
## List Operations
### pop()

```
[3, 14, 0, 5, 8]
```

```
my_list.pop(3) ⟶ 1
my_list.pop(3) ⟶ 14
```

## List Manipulation
## List Operations

**+**

```
[3, 14, 0, 5, 8]
```

```
my_list_2 = [10, 9, 8, 7]
my_list = my_list + my_list_2
```

## List Manipulation
## List Operations

**+**

```
[3, 14, 0, 5, 8, 10, 9, 8, 7]
```

```
my_list_2 = [10, 9, 8, 7]
my_list = my_list + my_list_2
```

## List Manipulation
## List Operations
### sort()

```
[3, 14, 0, 5, 8, 10, 9, 8, 7]
```

```
my_list.sort()
```

## List Manipulation
## List Operations
### sort()

```
[0, 3, 5, 7, 8, 8, 9, 10, 14]
```

```
my_list.sort()
```

## List Manipulation
## List Operations
### len()

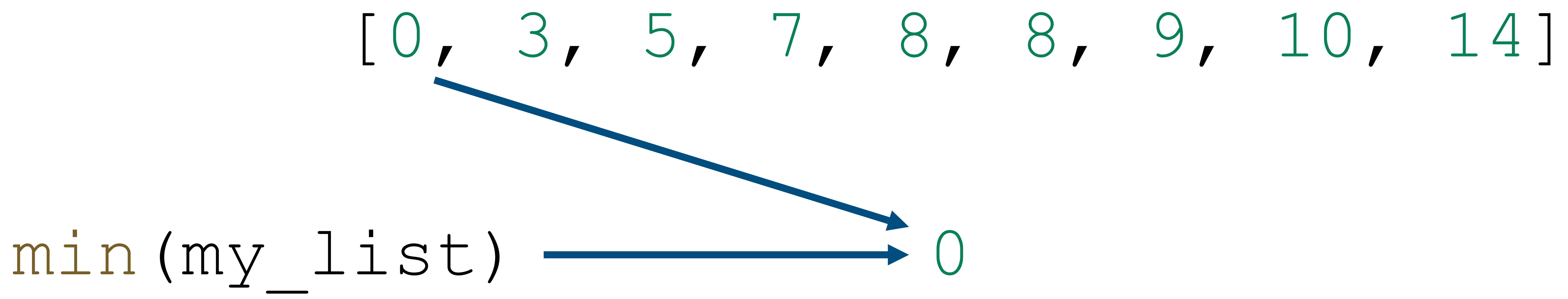
$$\underbrace{\texttt{[0, 3, 5, 7, 8, 8, 9, 10, 14]}}_{9}$$

$$\texttt{len(my\_list)} \longrightarrow 9$$

## List Manipulation
## List Operations
### max()

$$\begin{tikzcd}
 & & \texttt{[0, 3, 5, 7, 8, 8, 9, 10, 14]} \arrow[dl] \\
\texttt{max(my\_list)} \arrow[r] & \texttt{14} &
\end{tikzcd}$$

## List Manipulation
## List Operations
### min()



```
[0, 3, 5, 7, 8, 8, 9, 10, 14]

min(my_list)  →  0
```

## List Manipulation

Indexing
List Operations
Listcomp
String/list Interop
Multidimensional Lists

## List Manipulation
## Listcomp

### Shorthand for “for” loops

```
new_list = [expression for object in iteration]
```

```
[obj1, obj2, obj3, obj4, obj5, obj6, obj7 ...]
```

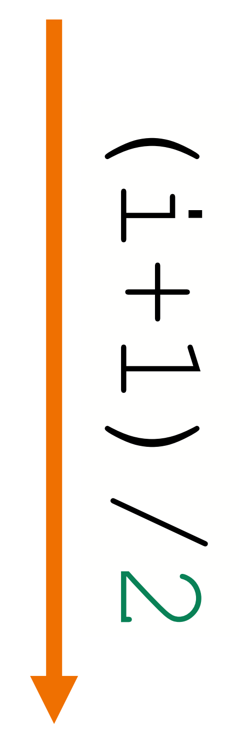
$$\Big\downarrow \text{expression}$$

```
[new1, new2, new3, new4, new5, new6, new7 ...]
```

## List Manipulation
## Listcomp

```
new_list = [(i+1)/2 for i in range(7)]
```

```
[0,    1,    2,    3,    4,    5,    6]
```



```
[0.5,  1.0,  1.5,  2.0,  2.5,  3.0,  3.5]
```

## List Manipulation

Indexing
List Operations
Listcomp
String/list Interop
Multidimensional Lists

## List Manipulation
## String/list Interop
### join()

List of strings

```
my_list = [str1, str2, str3]

separator.join(my_list)
```

## List Manipulation
## String/list Interop
## join()

List of strings

```
my_list = [str1, str2, str3]

separator.join(my_list)
```

Final String

```
str1 separator str2 separator str3
```

## List Manipulation
## String/list Interop
### join()

```
my_list = ["Hello,", "my", "name", "is", "Bob!"]

' '.join(my_list)
```

## List Manipulation
## String/list Interop
### join()

```
my_list = ["Hello,", "my", "name", "is", "Bob!"]

' '.join(my_list)
```

```
"Hello,"my"name"is"Bob!"
```

## List Manipulation

Indexing

List Operations

Listcomp

String/list Interop

Multidimensional Lists

## List Manipulation
## Multidimensional Lists

**A list inside a list [inside a list inside …]**

## List Manipulation
## Multidimensional Lists

**A list inside a list [inside a list inside ...]**

```
my_list[i]

my_list[i][j]
```

```
   my_list
[[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]]
```

## That was a lot!

**Let’s get to the lab!**