AIBridge

Lecture 2

Lecture Outline

I/O

List Manipulation

Object Oriented Programming

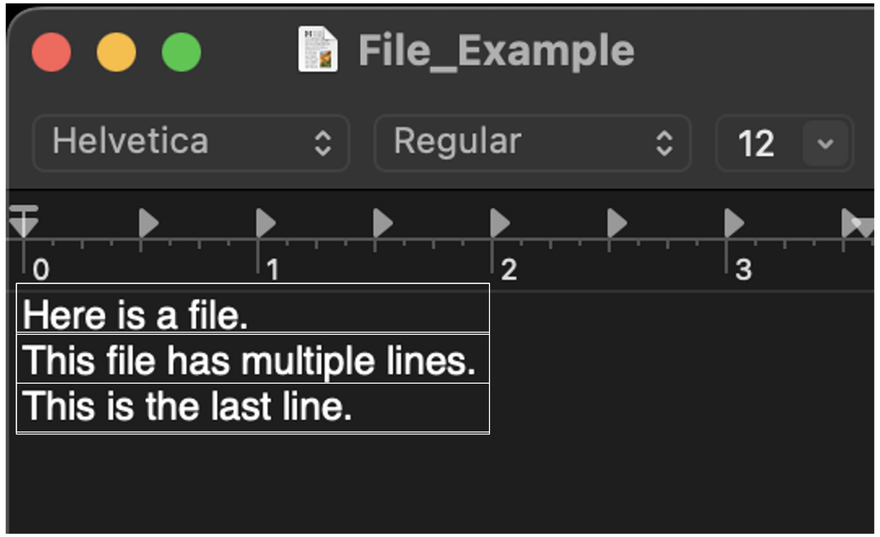
I/O
Standard Input

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."""
```

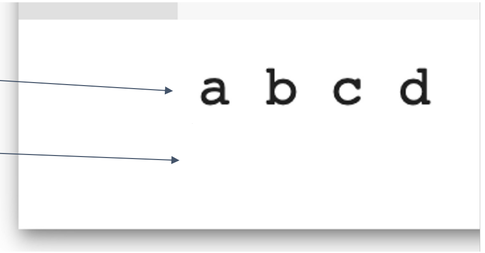
```
"Here is a file."
"This file has multiple lines."
"This is the last line."
```

```
["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')
```



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

`write()`

Always close file

Note: This removes any existing file with that name

Lecture Outline

I/O

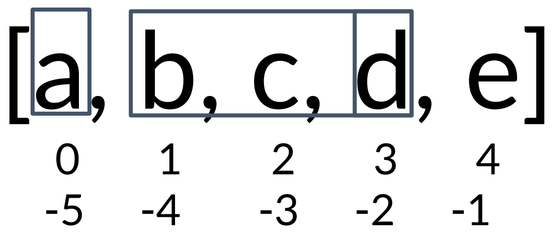
List Manipulation

Object Oriented Programming

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 print?

```
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]

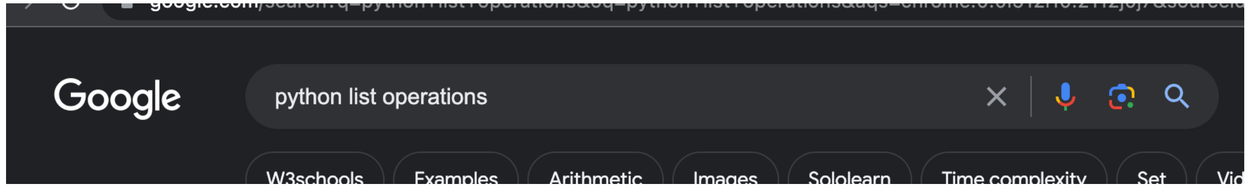
List Manipulation

Indexing
List Operations
Listcomp
String/list Interop
Multidimensional Lists

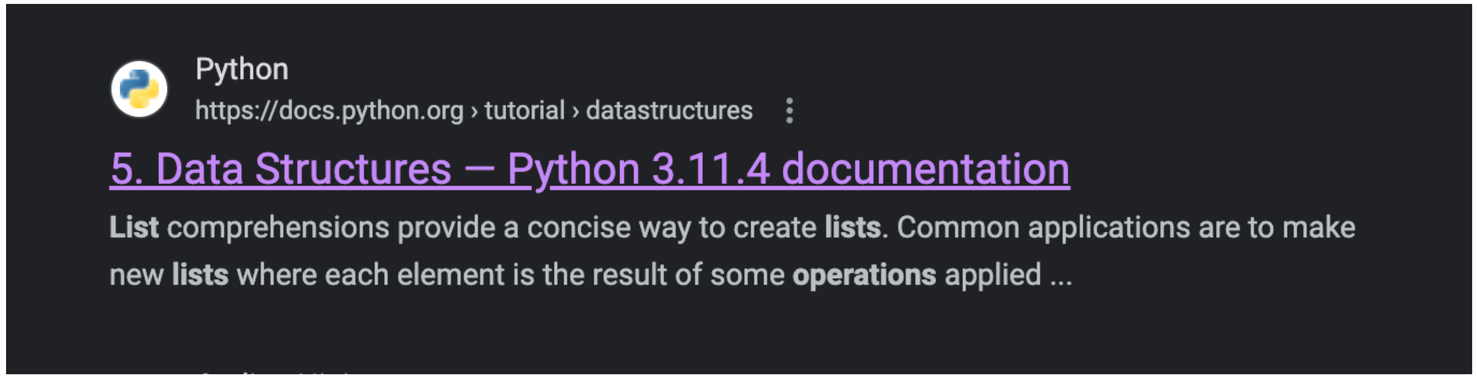
List Manipulation
List Operations

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

1



2



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
pop()

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

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

List Manipulation
List Operations
pop()

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

```
my_list.pop(3) ⟶ 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()

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

```
len(my_list) ──────────► 9
```

List Manipulation
List Operations
max()

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

max(my_list) → 14
```

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 ...]
```

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]
```

(i+1)/2

```
[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
String/list Interop
split()

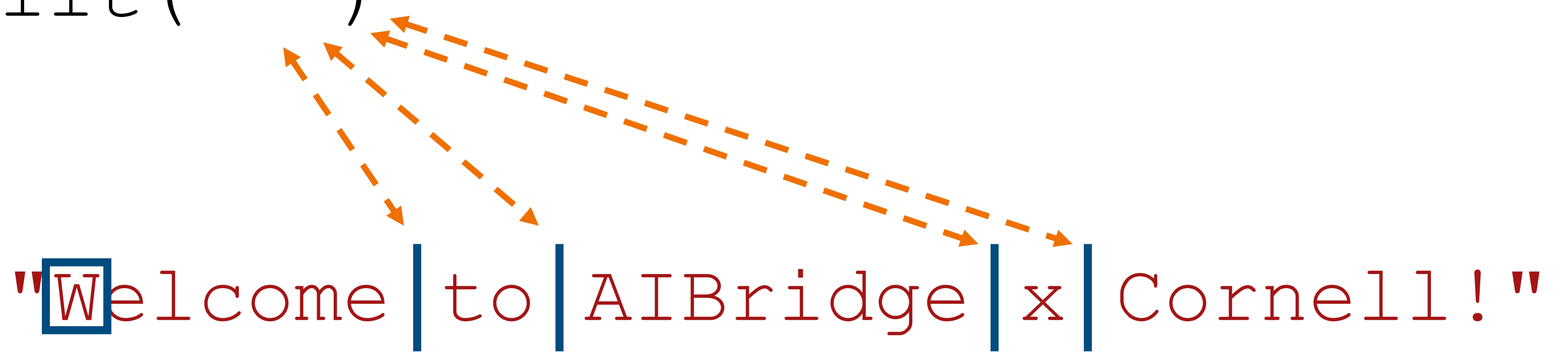
```
my_string = "Welcome to AIBridge x Cornell!"

my_string.split(' ')
```

List Manipulation
String/list Interop
split()

```
my_string = "Welcome to AIBridge x Cornell!"

my_string.split(' ')
```



List Manipulation
String/list Interop
split()

```
my_string = "Welcome to AIBridge x Cornell!"
my_string.split(' ')
```

```
"Welcome"to"AIBridge"x"Cornell!"
```

List Manipulation
String/list Interop
split()

```
my_string = "Welcome to AIBridge x Cornell!"

my_string.split(' ')

→ ["Welcome", "to", "AIBridge", "x", "Cornell!"]
```

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[0]

my_list[0][0]
```

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

Lecture Outline

I/O

List Manipulation

Object Oriented Programming

Object Oriented Programming

A blueprint

Class

Instructions for data

Methods / Functions

Usually deal
with data

Object Oriented Programming

A blueprint

Class

Instructions for data

Methods / Functions

Specific to class

Object Oriented Programming

A blueprint

Class

- Instructions for data
- Methods / Functions

Instance built from blueprint

Object

- Actual data
- Ability to use functions

Object Oriented Programming

A blueprint

List

- "Make space for elements"
- append(), pop(), indexing, etc.

Instance built from blueprint

My_list

- [5, 6, 1, 3, 0, -1]
- Ability to use functions

Data can be other object instances

Object Oriented Programming

List

- **"Make space for elements"**
- **append(), pop(), indexing, etc.**

My_list

- **[5, 6, 1, 3, 0, -1]**
- **Ability to use functions**

```
My_list.append(1)
```

Object Oriented Programming

List

- **"Make space for elements"**
- **append(), pop(), indexing, etc.**

My_list

- **[5, 6, 1, 3, 0, -1, 1]**
- **Ability to use functions**

```
My_list.append(1)
```

That was a lot!

Let’s get to the lab!