香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Introduction to Computer Science: Programming Methodology

Lecture 4 Function

Prof. Yunming XIAO

School of Data Science

Outline

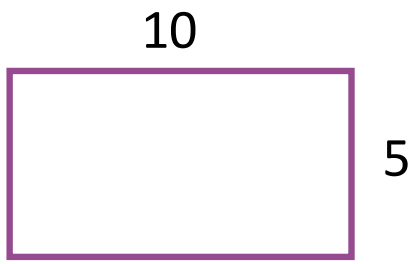
- Function Basics
- String Methods
- File Operations

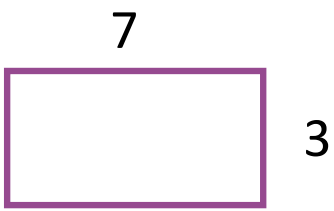
Outline

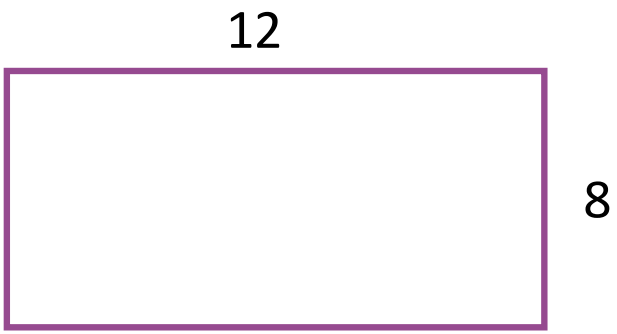
- **Function Basics**
- String Methods
- File Operations

Exercise

Imagine you need to calculate the area of multiple rectangles throughout your program. Print the areas of the following rectangles.







A Python script

```python
# Calculate area of rectangle 1
length1 = 10
width1 = 5
area1 = length1 * width1
print(f"Area of rectangle 1: {area1}")

# Calculate area of rectangle 2
length2 = 7
width2 = 3
area2 = length2 * width2
print(f"Area of rectangle 2: {area2}")

# Calculate area of rectangle 3
length3 = 12
width3 = 8
area3 = length3 * width3
print(f"Area of rectangle 3: {area3}")
```

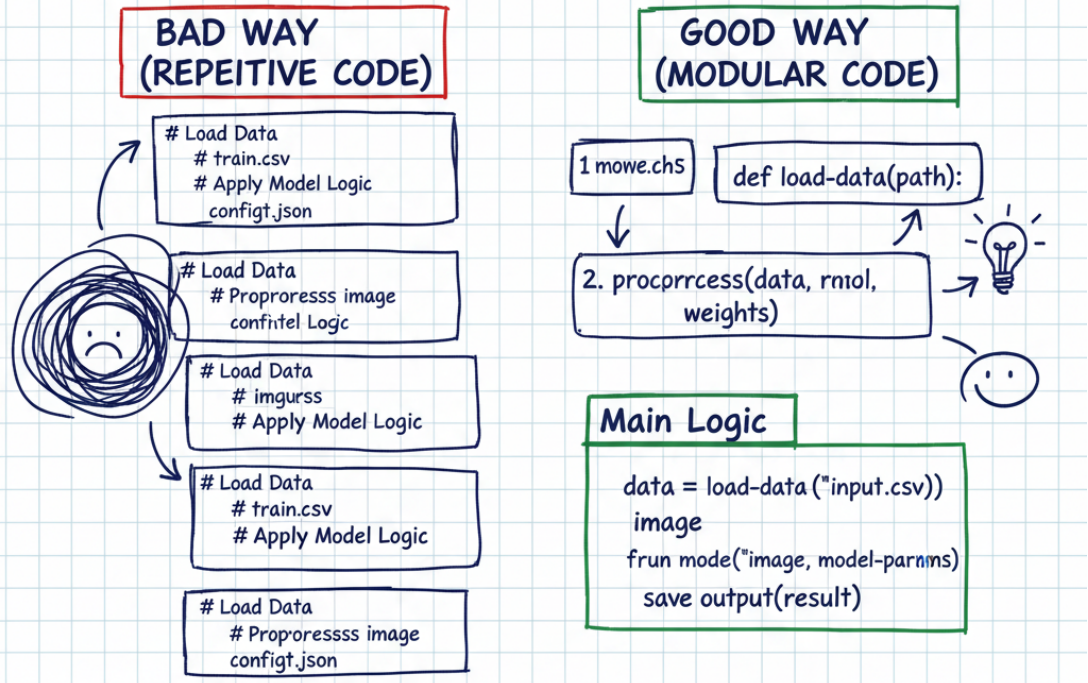
Same Pattern Applies!

Can we define a template and reuse it?

A better solution

```
def calculate_rectangle_area(length,width):
    return length * width

# Calculate area of rectangle 1
area1 = calculate_rectangle_area(10, 5)
print(f"Area of rectangle 1: {area1}")

# Calculate area of rectangle 2
area2 = calculate_rectangle_area(7, 3)
print(f"Area of rectangle 2: {area2}")

# Calculate area of rectangle 3
area3 = calculate_rectangle_area(12, 8)
print(f"Area of rectangle 3: {area3}")
```

PYTHON FUNCTIONS (CONCEPT)



* Crucial for:

✓ Reganizality (DRY - Don't Repeait Yourrself)

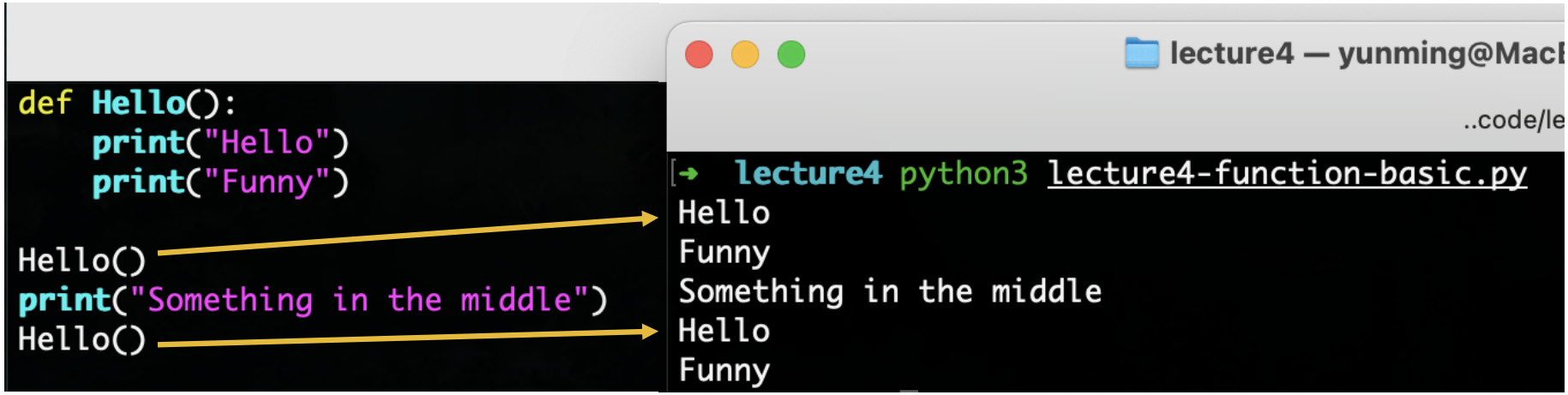
✓ Modaulliity & Collaboration

• Abstration (Hide Complenty)

Stored (and reused) steps

Program

Outputs



- This reusable paragraph of code is called a function

Python functions

- In Python, a function is some reusable code which can take arguments as input, perform some computations, and then output some results

- Functions are defined using reserved word def

- We call/invoke a function by using the function name, parenthesis and arguments in an expression

Python functions

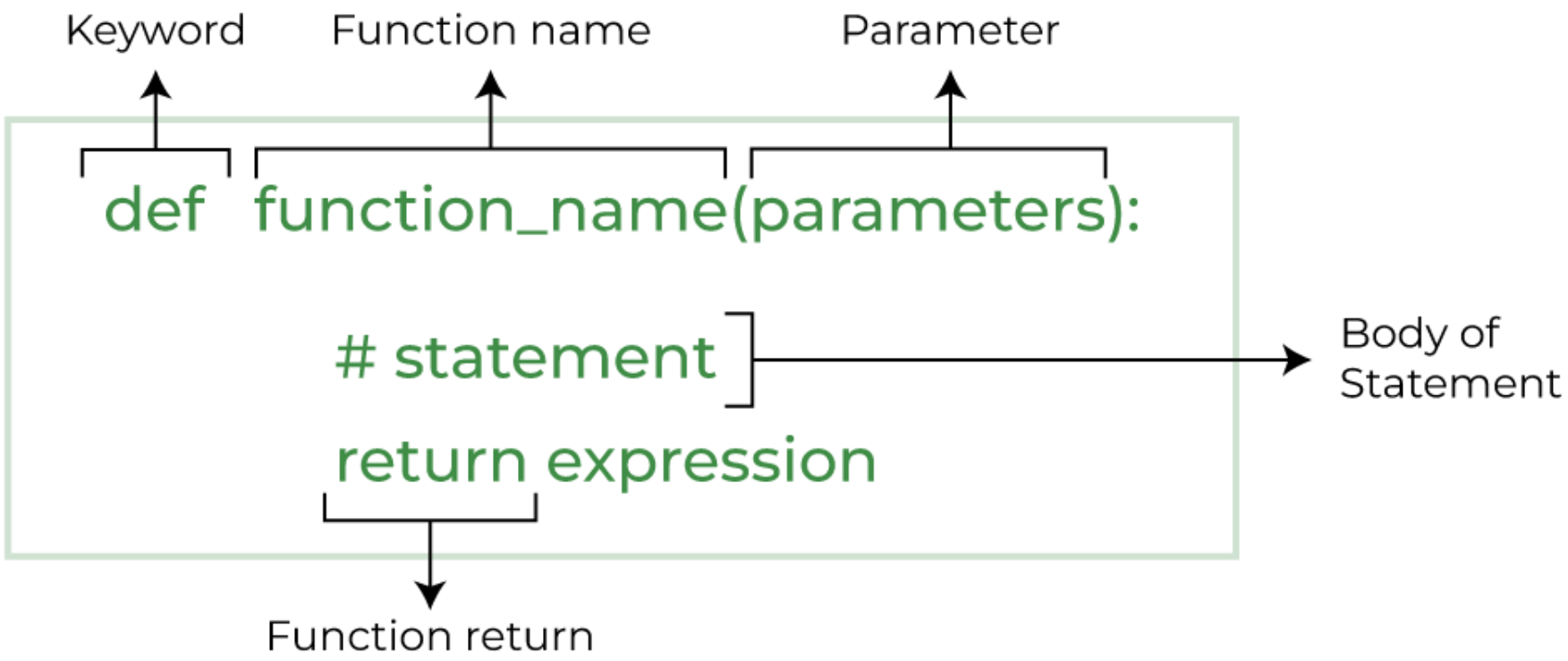
- There are (mainly) two types of functions in Python
  - Built-in functions which are part of Python, such as print(), int(), float(), etc
  - Functions that we define ourselves and then use

- The names of built-in functions are usually considered as new reserved words, i.e. we do not use them as variable names

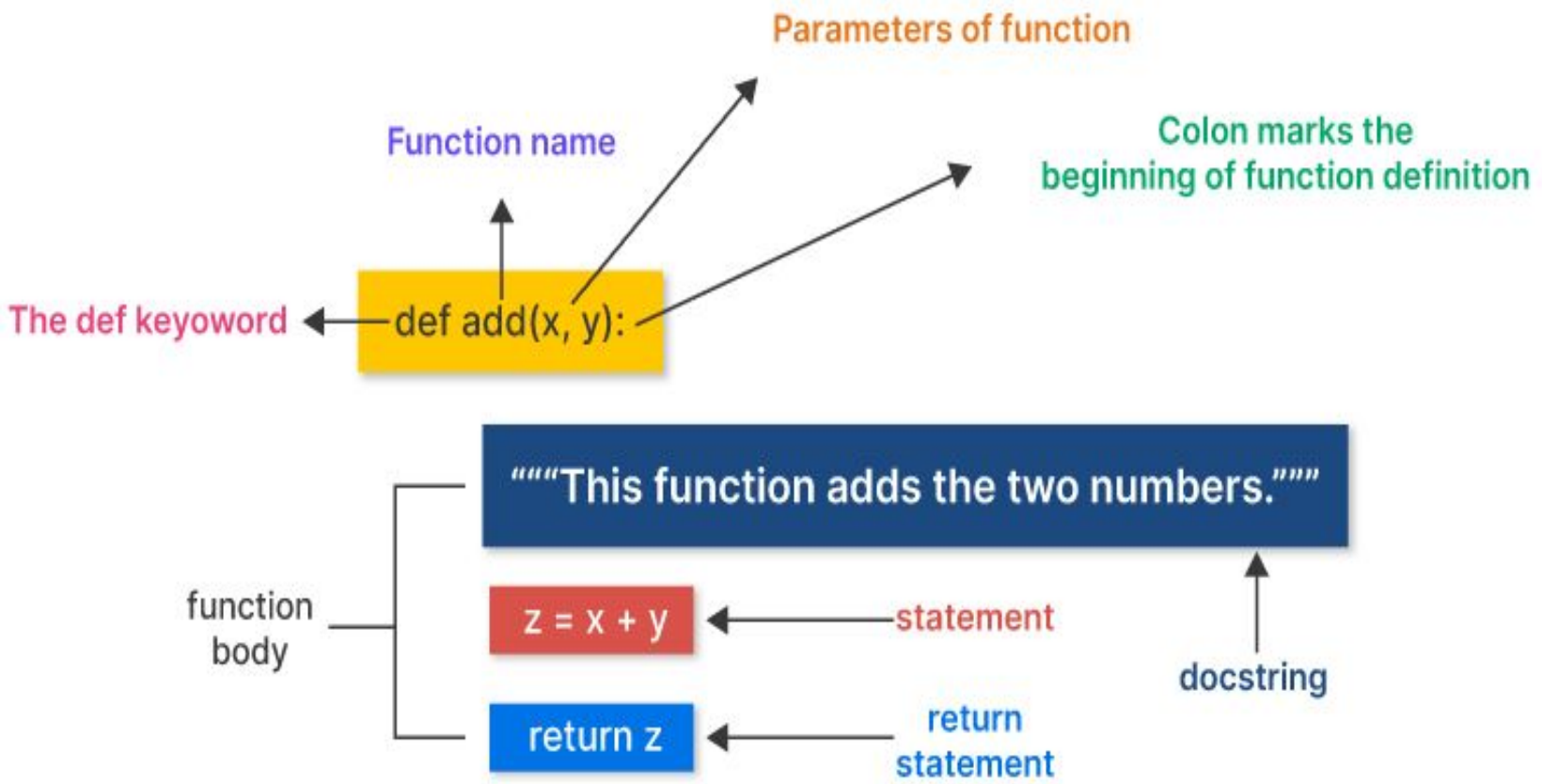
Building our own functions

- We create a new function using the def key word, followed by optional parameters in parenthesis
- We indent the body of the function
- This defines the function, but does not execute the body of the function

Building our own functions



A sample code



Argument

- An argument is a value we pass into the function as its input when we call the function

- We use arguments so we can direct the function to do different kinds of work when we call it at different times

- We put the argument in parenthesis after the name of the function

```
big = max("I am the one")
```

argument

Parameters

- A parameter is a variable which we use in the function definition that is a 'handle' that allows the code in the function to access the arguments for a particular function invocation

Return values

- Often a function will take its arguments, do some computation and return a value to be used as the value of the function call in the calling expression. The return keyword is for this purpose.

```
>>> def greet():
...     return "Hello"
...
>>> print(greet(), "Glenn")
Hello Glenn
>>> print(greet(), "Sally")
Hello Sally
```

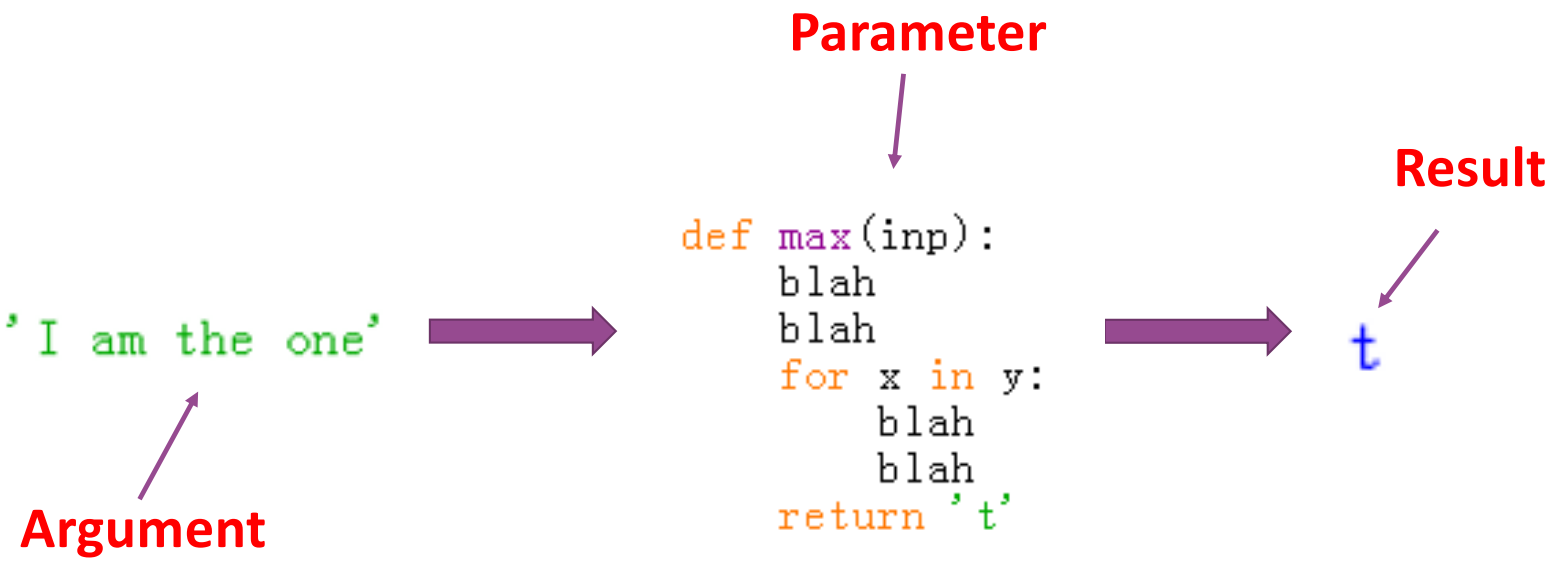
Return values

- A fruitful function is one that produces a result (or return value)

- The return statement ends the function execution and ‘sends back’ the result of the function

Argument, parameter, and result

```
>>> big = max("I am the one")
>>> print(big)
t
```



Multiple arguments/parameters

- We can define more than one parameter in a function definition
- We simply add more arguments when we call the function
- We match the number and order of arguments and parameters

```
def add_two(a, b):
    total = a + b
    return total

x = add_two(3, 5)
print(x)
```

Default argument

- Python allows to define functions with default argument values
- The default argument values will be passed to the function, when it is invoked without arguments

```
def print_area(width=1, height=2):
    area = width * height
    print("width:", width, "\theight:", height, "\tarea:", area)

print_area() # Default arguments, width = 1 and height = 2
print_area(4, 2.5) # Positional arguments, width  = 4 and height = 2.5
print_area(height=5, width=3) # Keyword arguments
print_area(width=1.2) # Default height = 2
print_area(height=6.2) # Default width = 1
```

Return multiple values

- Python allows a function to return multiple values

- The sort function returns two values; when it is invoked, you need to pass the returned values in a simultaneous assignment

```
def sort(num1, num2):
    if num1 < num2:
        return num1, num2
    else:
        return num2, num1

n1, n2 = sort(3, 2)
print("n1 is", n1)
print("n2 is", n2)
```

Void functions

- When a function does not return a value, it is called a “void” function
- Functions that return values are “fruitful” functions
- Void functions are “not fruitful”
- “fruitful” != useful

Scope of variables

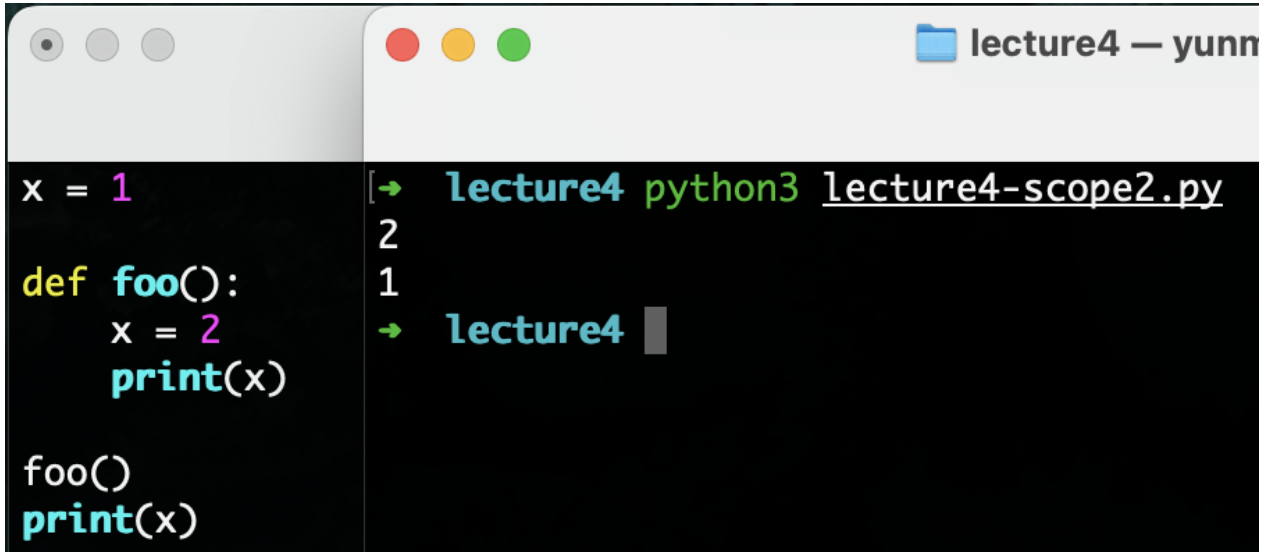
- The scope of a variable is the part of program where this variable can be accessed
- A variable created inside a function is referred to as a local variable
- Global variables are created outside all functions and are accessible to all functions in their scope

```
global_var = 1

def foo():
    local_var = 2
    print(global_var)
    print(local_var)

foo()
print(global_var)
print(local_var)
```

Error!

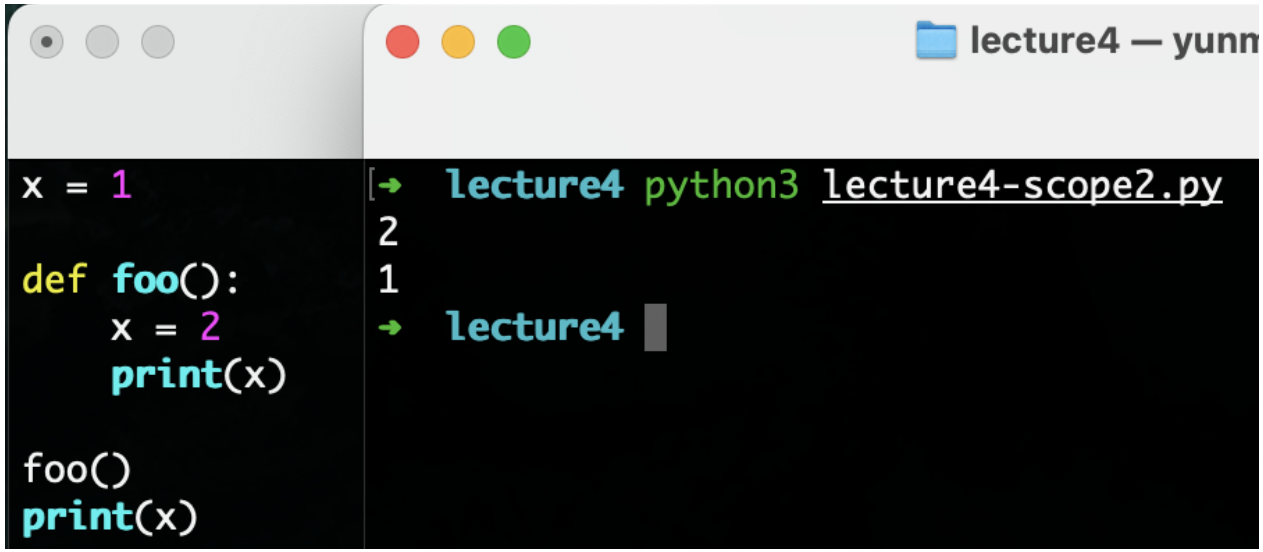
Scope of variables

- Different variables may share a name if they have different scopes



Scope of variables

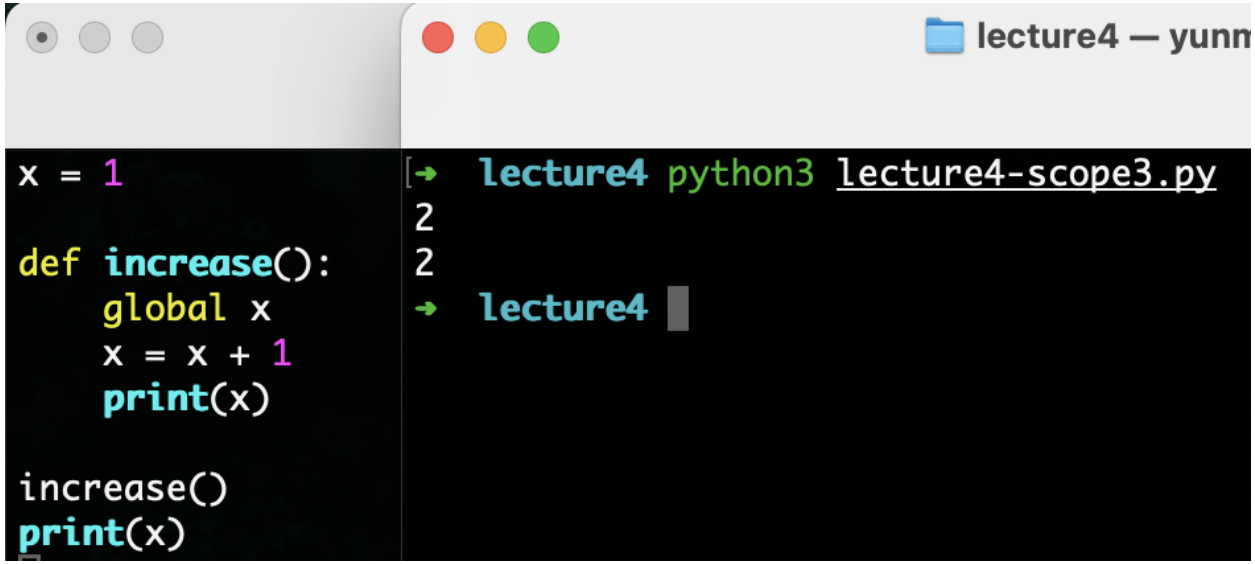
- Different variables may share a name if they have different scopes



Scope of variables

- In a function, you can use keyword global to specify that a variable is a global variable

- Be very careful when define and use global variable



To function or not function…

- Organize your code into paragraphs - capture a complete thought and name it
- Don’t repeat yourself – name it to work once and reuse it
- If something goes too complex, break up them into several blocks, and put each of them into a function
- Make a library of common stuffs that you do over and over again – perhaps share with other people

Practice

- Write a function to instruct the user to input the working hours and hourly rate, and then return the salary. If the working hours exceed 40 hours, then the extra hours received 1.5 times pay.

Practice

- Write a function to get the tax for a given salary (excluded tax-free parts) in a year.
  - If the salary is less than 36000. It takes 3%
  - If the salary is between 36000 and 144000, it takes **3**% for the part less than 36000; **10**% for the part between 36000 and 144000.
  - ….

1	0 - 36000	3
2	36000. 01 - 144000	10
3	144000. 01 - 300000	20
4	300000. 01 - 420000	25
5	420000. 01 - 660000	30
6	660000. 01 - 960000	35
7	960000. 01 - ∞	45

Break

Outline

- Function Basics
- **String Methods**
- File Operations

String type

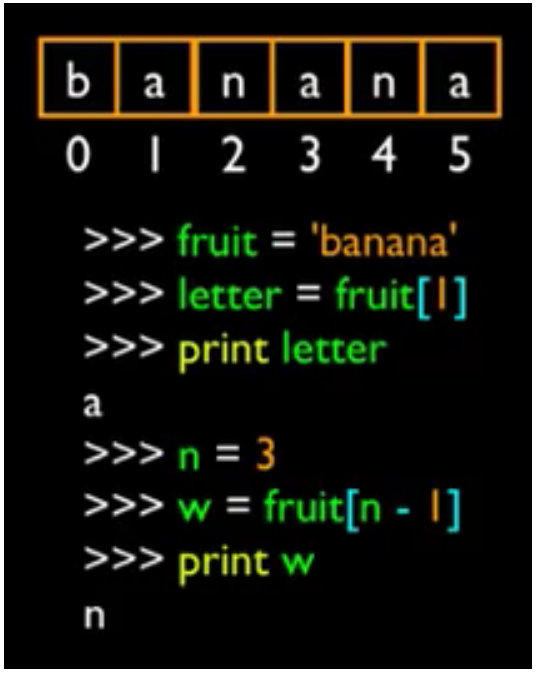
- A string is a sequence of characters
- A string literal uses quotes ‘’ or “”
- For strings, + means “concatenate”
- When a string contains numbers, it is still a string
- We can convert numbers in a string into a number using int() or float()

Reading and converting

- We prefer to read data in using strings and then parse and convert the data as we need
- This gives us more control over error situations and/or bad user inputs
- Raw input numbers must be converted from strings

Looking inside the strings

- We can get any character in a string using an index specified in square brackets
- The index value must be an integer which starts from zero
- The index value can be an expression



Index out of range

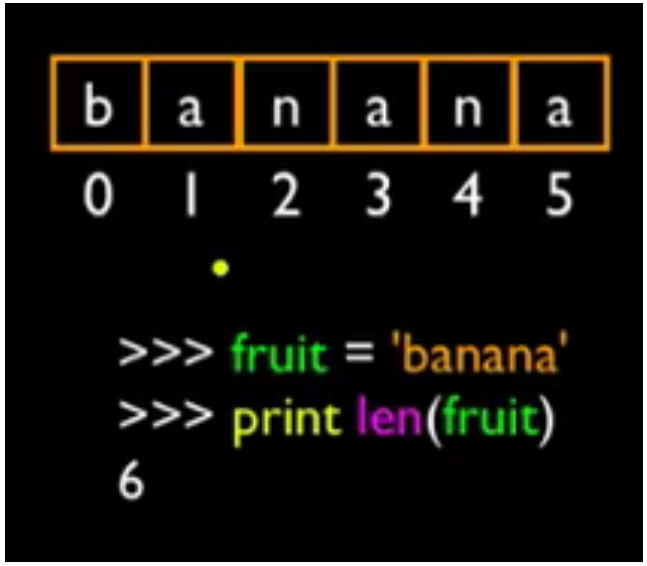
- You will get a Python error if you attempt to index beyond the end of a string

- Be careful when specifying an index value

```
>>> name = "Alice"
>>> name[6]
Traceback (most recent call last):
  File "<python-input-1>", line 1, in <module>
    name[6]
    ~~~~^^^
IndexError: string index out of range
>>>
```

Strings have length

- There is a built-in function len() which gives us the length of a string



len()

len(object)

- Returns the **length** of the given string, array, list, tuple, dictionary, or any other iterable or container object.
- The type of the return value is an integer that represents the number of elements in this iterable.

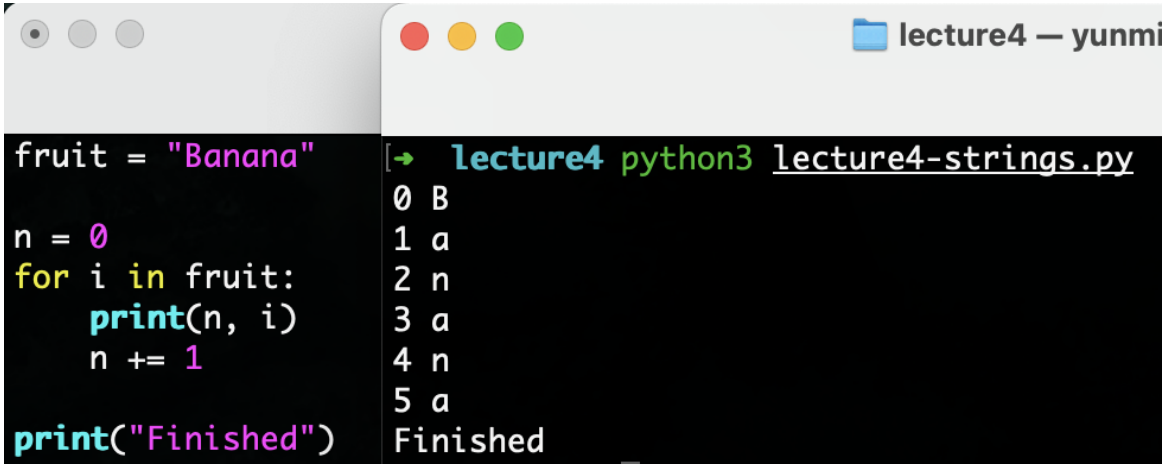
➡ *Calls* `object.__len__()`

```
>>> friends = ['Alice', 'Bob', 'Carl', 'Ann']
>>> len(friends)
4
>>> friends.extend([1, 2, 3])
>>> len(friends)
7
```

Strings have length

- Using a for statement, we can easily loop through each character in a string

- String is essentially a list in Python



Practice

- Write a program to use a while statement together with len() function to loop through a given string.

Slicing strings

- We can also look at any continuous section of a string using colon operator

- The second number is one beyond the end of the slice – i.e. "up to but not including"

```
>>> s = "Monty Python"
>>> print(s[0:4])
Mont
>>> print(s[6:7])
P
>>> print(s[6:20])
Python
```

- If the second number is beyond the length of the string, it stops at the end

Slicing strings

- If we leave off the first or second number of the slice, it is assumed to be the beginning or end of the string respectively


```
>>> print(s[:6])
Monty
>>> print(s[3:])
ty Python
>>> print(s[:])
Monty Python
>>>
```

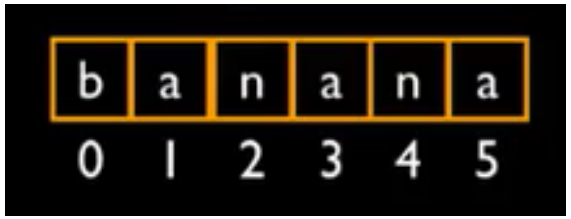
Keyword “in”

- The in keyword can be used to check whether one string is in another string
- The in expression is a logical expression and returns True or False
- It can be used in if or while statement

```
>>> fruit = "Banana"
>>> "n" in fruit
True
>>> "m" in fruit
False
>>> "nan" in fruit
True
```

Search a string

- We can use the find() function to search for a substring in a string
- find() finds the first occurrence of the target sub-string
- If the sub-string is not found, it returns -1
- Important: the string position starts from 0




```
>>> fruit = "Banana"
>>> pos = fruit.find("na")
>>> print(pos)
2
>>> pos = fruit.find("z")
>>> print(pos)
-1
```

Search and replace

- The replace() function is like a “search and replace” operation in a word processor

- It replaces all occurrences of the search string with the replacement string

```
>>> greet = "Hello, Bob"
>>> new_str = greet.replace("Bob", "Jane")
>>> print(new_str)
Hello, Jane
>>> new_str = greet.replace("o", "X")
>>> print(new_str)
HellX, BXb
>>> new_str = greet.replace("z", "X")
>>> print(new_str)
Hello, Bob
```

Stripping whitespace

- Sometimes we want to take a string and remove whitespaces at the beginning and/or end
- lstrip() and rstrip() to the left and right only
- strip() removes both beginning and ending whitespaces

```
>>> greet = " Hello Bob\n"
>>> greet.lstrip()
'Hello Bob\n'
>>> greet.rstrip()
' Hello Bob'
>>> greet.strip()
'Hello Bob'
```

Prefixes

- startswith() function checks whether a string is starting with a given string

```
>>> line = "Please submit your application"
>>> line.startswith("Please")
True
>>> line.startswith("p")
False
```

String library

```
>>> dir(fruit)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__'
, '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__', '__init__', '__in
it_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduc
e__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclass
hook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'f
ormat_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnu
meric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'part
ition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip
', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
>>>
```

https://docs.python.org/3/library/stdtypes.html#string-methods

PYTHON STRINGS (CONCEPT)

Essential Methods & Operations

Creation & Length

my string = "Hello, Python!
length = len)

→ length 14

Whitsspaice & More

text "hello world!"
I strip = texxrestrl)"

Replacing

"drist in rstrip)"

Case Conversion

upper str = my string.upper"
lower str = world.lower"

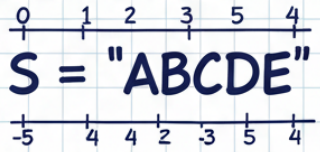
hello world!

Find & Check

index = my world.find("Py"
starts with = stacbstesh)
weights)

"hellothon in

Indexing & Slicing



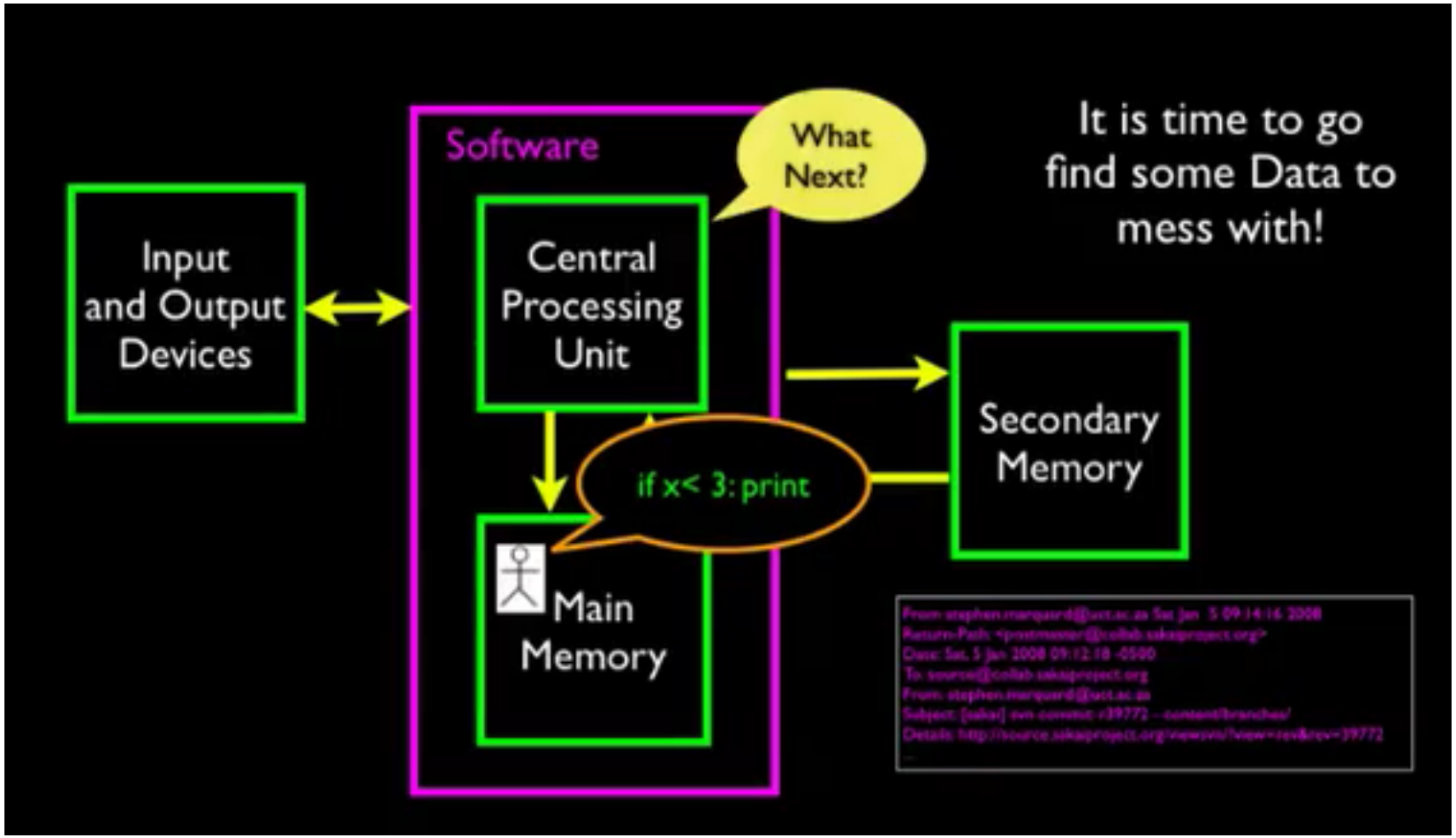
first	"A	slice1	slice2
last	"BCD"	BCD3	revers"
rave	–21"	"PDE"	"ED//BCA"

* Crucial for:

- len (s): Get length"): Use methods
- []in: Check Check for Slicing
- Abstration (Hide Complenty !

Outline

- Function Basics
- String Methods
- **File Operations**


Software
What Next?
It is time to go find some Data to mess with!
Input and Output Devices
Central Processing Unit
Secondary Memory
if x< 3: print
Main Memory

File processing

- A text file can be thought of as a sequence of lines

```
# Gmail web Start
216.239.38.125  chatenabled.mail.google.com
216.239.38.125  filetransferenabled.mail.google.com
216.239.38.125  gmail.com
216.239.38.125  gmail.google.com
216.239.38.125  googlemail.l.google.com
216.239.38.125  inbox.google.com
216.239.38.125  isolated.mail.google.com
216.239.38.125  m.gmail.com
216.239.38.125  m.googlemail.com
216.239.38.125  mail.google.com
216.239.38.125  www.gmail.com
# Gmail web End
```

Opening files

- Before we can read the contents of a file, we must tell Python which file we are going to work with and what we will do with that file

- This is done with the open() function

- Open() returns a "file handle" - a variable used to perform operations on files

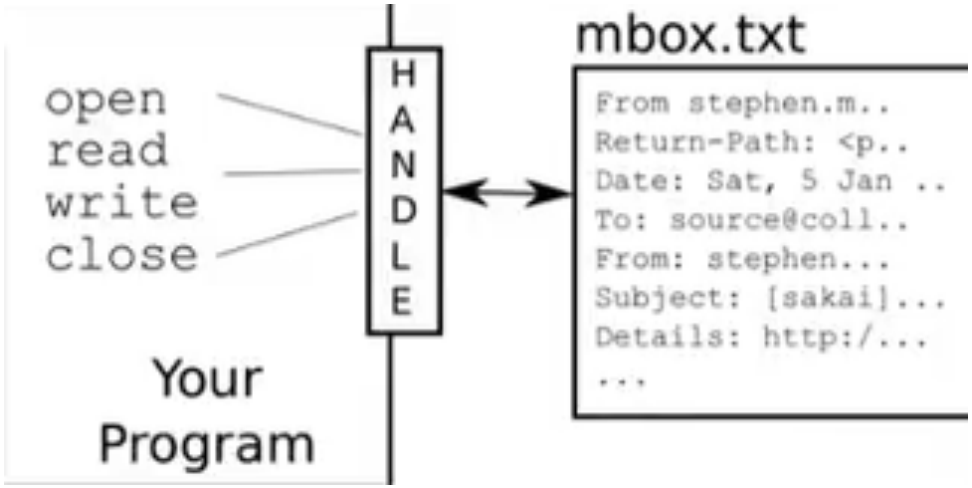
- Kind of like "File -> Open" in a word processor

open()

- handle = open(filename, mode)
- Returns a handle used to manipulate the file
- Filename is a string
- Mode is optional, use ‘r’ if we want to read the file, and ‘w’ if we want to write to the file **(‘r’, ‘w’, ‘a’**, ‘r+’, ‘w+’, ‘a+’, more for binary files …)

File handle

```
>>> file_handle = open("lecture4-strings.py", "r")
>>> print(file_handle)
<_io.TextIOWrapper name='lecture4-strings.py' mode='r' encoding='UTF-8'>
>>>
```



When files are missing

```
>>> file_handle = open("non-exist.py", "r")
Traceback (most recent call last):
  File "<python-input-74>", line 1, in <module>
    file_handle = open("non-exist.py", "r")
FileNotFoundError: [Errno 2] No such file or directory: 'non-exist.py'
>>>
```

The newline character

- We use a new character to indicate when a line ends called “newline”
- We represent it as ‘\n’ in strings
- Newline is still one character, not two

```
>>> s = "Hello\nWorld"
>>> s
'Hello\nWorld'
>>> print(s)
Hello
World
```

File processing

- A text file can be thought of as a sequence of lines
- A text file has newline at the end of each line

```
# Gmail web Start\n
216.239.38.125  chatenabled.mail.google.com\n
216.239.38.125  filetransferenabled.mail.google.com\n
216.239.38.125  gmail.com\n
216.239.38.125  gmail.google.com\n
216.239.38.125  googlemail.l.google.com\n
216.239.38.125  inbox.google.com\n
216.239.38.125  isolated.mail.google.com\n
216.239.38.125  m.gmail.com\n
216.239.38.125  m.googlemail.com\n
216.239.38.125  mail.google.com\n
216.239.38.125  www.gmail.com\n
# Gmail web End\n
```

File handle as a sequence

- A file handle open for read can be treated as a sequence of strings where each line in the file is a string in the sequence

- We can use the for statement to loop through a sequence

```
file_hand = open("lecture4-strings.py", "r")

for line in file_hand:
    print(line)

file_hand.close()
```

Practice

- Write a program to open a file and count how many lines are included in this file

Reading the whole file

- We can also read the whole file into a single string

```
file_hand = open("lecture4-strings.py", "r")

all_text = file_hand.read()
print("The length of the file:", len(all_text))
print("The first 20 characters of the file:", all_text[:20])
```

Practice

- Write a program to open a file and read all texts with read() and count how many lines are included in this file

Writing to a file

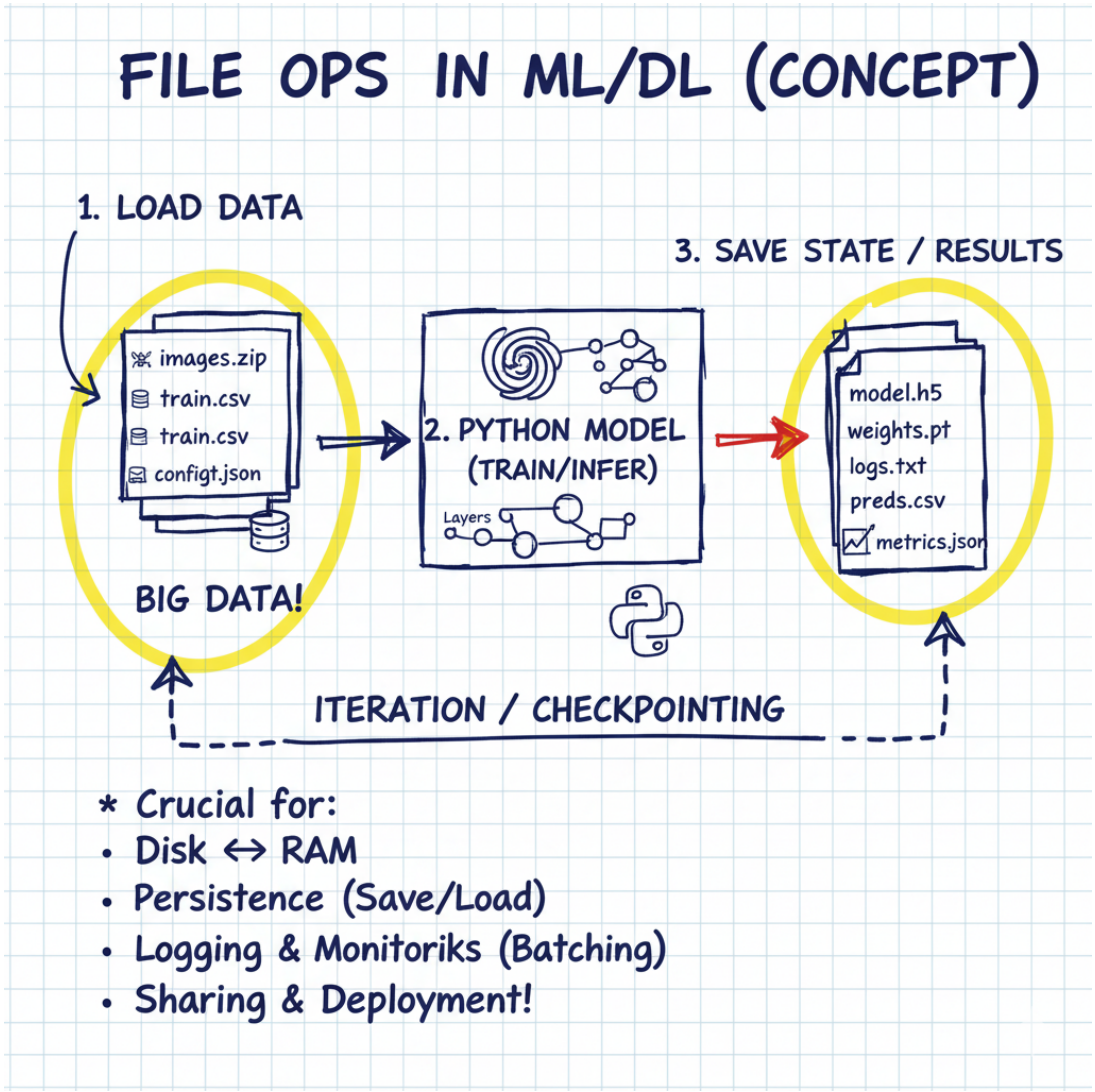
- To write a file, use the open() function with ‘w’ argument

- Use the write() method to write to the file

```
file_hand = open("test.txt", "w")
file_hand.write("The first line\n")
file_hand.write("The second line\n")
file_hand.write("The third line\n")
file_hand.close()
```

Practice

Read a file and make all letters be lower-cased (to a new file)


FILE OPS IN ML/DL (CONCEPT)
1. LOAD DATA
3. SAVE STATE / RESULTS
images.zip
train.csv
train.csv
configt.json
BIG DATA!
2. PYTHON MODEL
(TRAIN/INFER)
Layers
model.h5
weights.pt
logs.txt
preds.csv
metrics.jsor
ITERATION / CHECKPOINTING
* Crucial for:
• Disk ↔ RAM
• Persistence (Save/Load)
• Logging & Monitoriks (Batching)
• Sharing & Deployment!

Thanks