The Python Programming Language (Part 3)

CST 205

Review of Python dictionaries

- A Python dictionary is a way to store data using key, value pairs.

```python
csumb_dictionary = {
    'year_founded' : 1994,
    'num_students' : 7_200,
    'first_gen_percent' : 65,
    'location' : 'Marina, California'
}

csumb_dictionary['num_students']
```

zip()

- Like a zipper for clothing!
    - Can create a dictionary from two lists using zip:

```
hex_values = ['A', 'B', 'C', 'D', 'E', 'F']

decimal_values = [10, 11, 12, 13, 14, 15]

hex_lookup = dict(zip(hex_values, decimal_values))
```

Working with files

- Basic input and output of data is a major requirement of any programming language.
- Need to consider how your programs will interact with users and with data stored in files.
- Text files are the workhorses of programming when it comes to saving data.

File objects

- In Python, `open()` returns a so-called **file object**.
- Think of a file object as Python’s representation of a file
  - Exposes methods such as `read()` and `write()`

Modes

- Today, we will use **`open()`** with two arguments:
    - the path to the file
    - the **mode**
- The modes we will use today are **`'r'`** (read mode) and **`'b'`** (binary mode).

`with` keyword

- We will use `with` when dealing with file objects
- `with` makes sure that the file is properly closed after we are finished with it.

Create a new file and write to it

```
with open('tmp.txt', 'w') as my_file:
    my_file.write('hello, world')
```

Take a look in your working directory.

Do you see a file called `tmp.txt`?

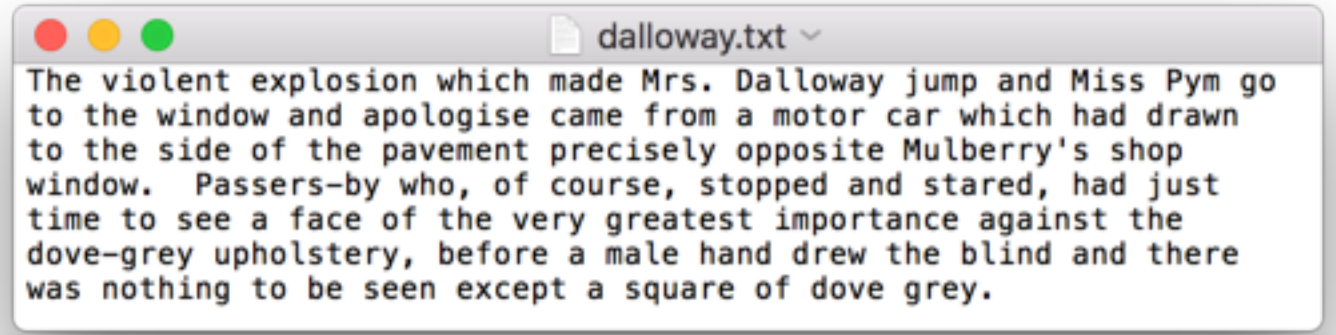
Open it and inpsect the contents.

Read in the file that we just created

```python
with open("tmp.txt", "r") as my_file:
    read_data = my_file.read()

print(read_data)
```

Opening a file with multiple lines



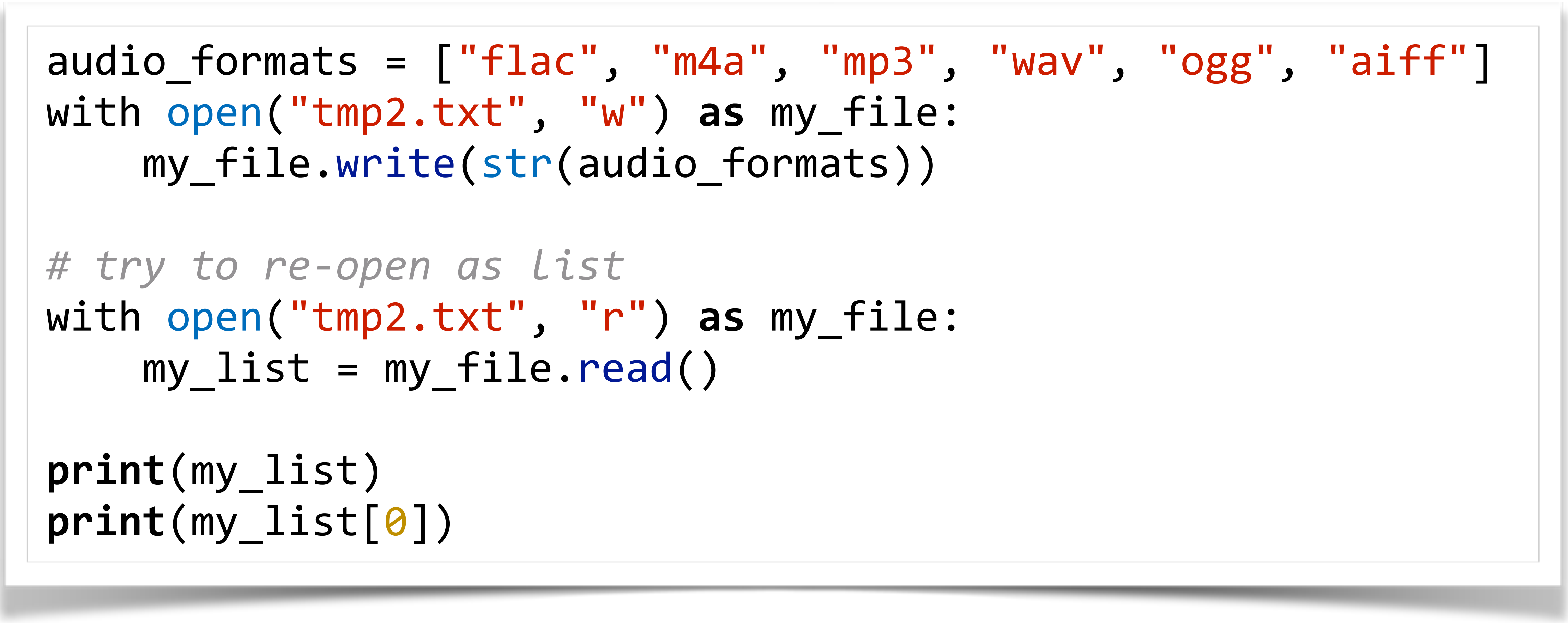
```
The violent explosion which made Mrs. Dalloway jump and Miss Pym go
to the window and apologise came from a motor car which had drawn
to the side of the pavement precisely opposite Mulberry's shop
window.  Passers-by who, of course, stopped and stared, had just
time to see a face of the very greatest importance against the
dove-grey upholstery, before a male hand drew the blind and there
was nothing to be seen except a square of dove grey.
```

```python
with open("dalloway.txt", "r") as f:
    line_list = f.readlines()

for counter, line in enumerate(line_list):
    print(f'Line {counter}: {line}')
```

Store Python list in a file

- From what we know so far, we can store it as a string.



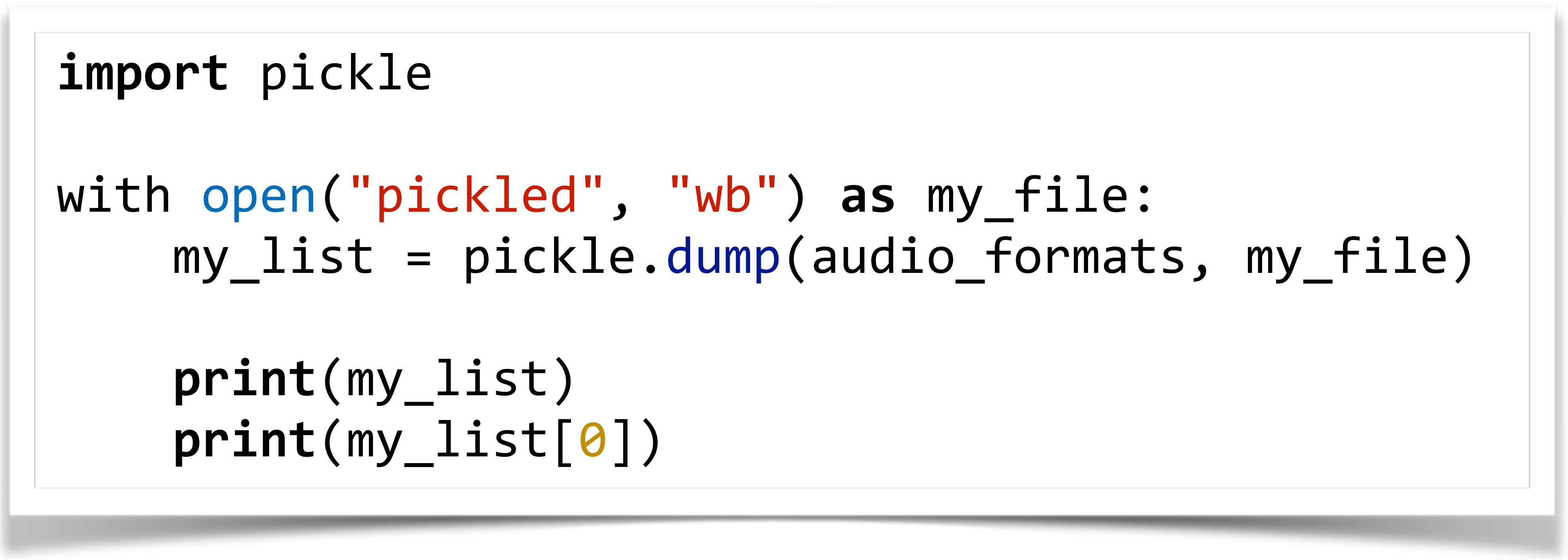
```python
audio_formats = ["flac", "m4a", "mp3", "wav", "ogg", "aiff"]
with open("tmp2.txt", "w") as my_file:
    my_file.write(str(audio_formats))

# try to re-open as list
with open("tmp2.txt", "r") as my_file:
    my_list = my_file.read()

print(my_list)
print(my_list[0])
```

Serializing

- Saving data structures to a file is called **serializing**.
- Python provides the **pickle** module to save and restore any object in a special binary format.



```
import pickle

with open("pickled", "wb") as my_file:
    my_list = pickle.dump(audio_formats, my_file)

    print(my_list)
    print(my_list[0])
```

Adding functionality with Python packages

- A **module** is a file containing Python definitions and statements.
- A Python **package** is a directory of one or more Python modules.
- The Python Packaging Authority (**PyPA**) is a working group that maintains many of the relevant projects in Python packaging.
  - PyPA's recommended tool for installing Python packages is `pip`
  - The general format is: `pip install` ***`PackageName`***

Virtual Environments

- The Python **`venv`** module allows you to create an isolated Python development space.
- Helps to alleviate confusion with different versions of Python.
- Makes life easier when installing Python packages
- Task 1 of Lab 4 walks you through the process of setting up a virtual environment.
- Python venv documentation