ITP 115 – Programming in Python

Pickling

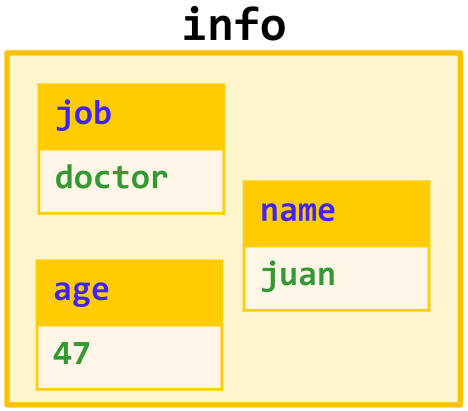
Recall:Text Files Are Great

- Great for storing **simple** information like strings (or ints we can convert to strings)
- They are cross-platform
- They are easy to use
  - Most operating systems come with basic tools to view and edit them

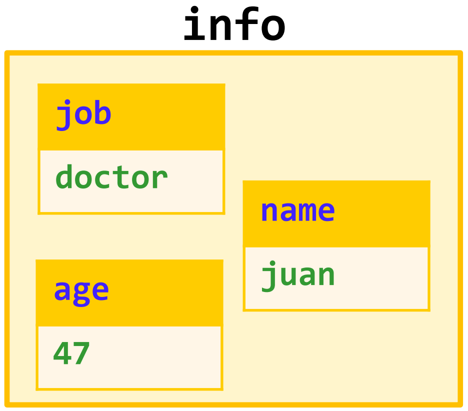
But…

- What if the data isn't simple?

Saving Complex Variables to Text Files



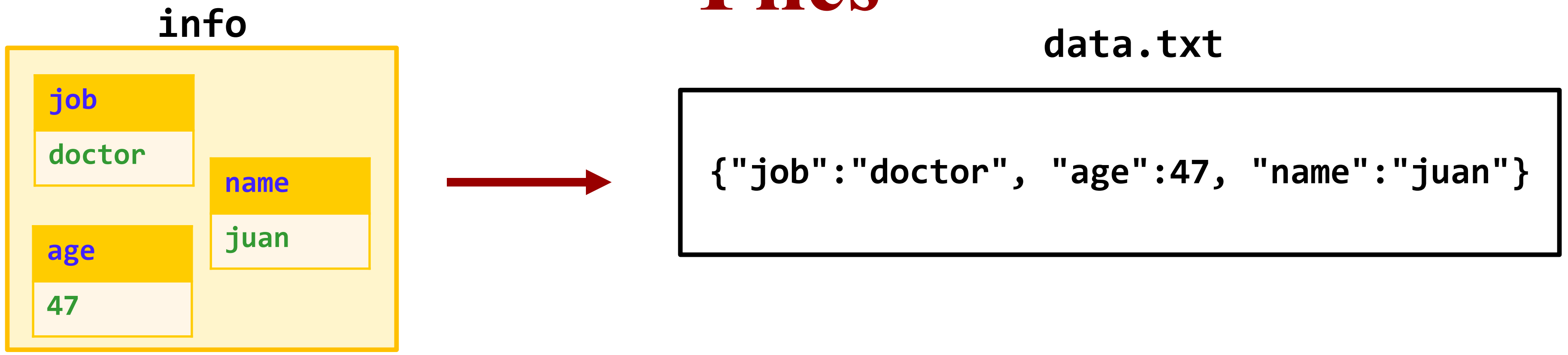
Saving Complex Variables to Text Files



```
fileOut = "data.txt"
print(info, file=fileOut)
fileOut.close()
```

- Does this work?

Saving Complex Variables to Text Files



```
fileOut = open("data.txt", "w)
print(info, file=fileOut)
fileOut.close()
```

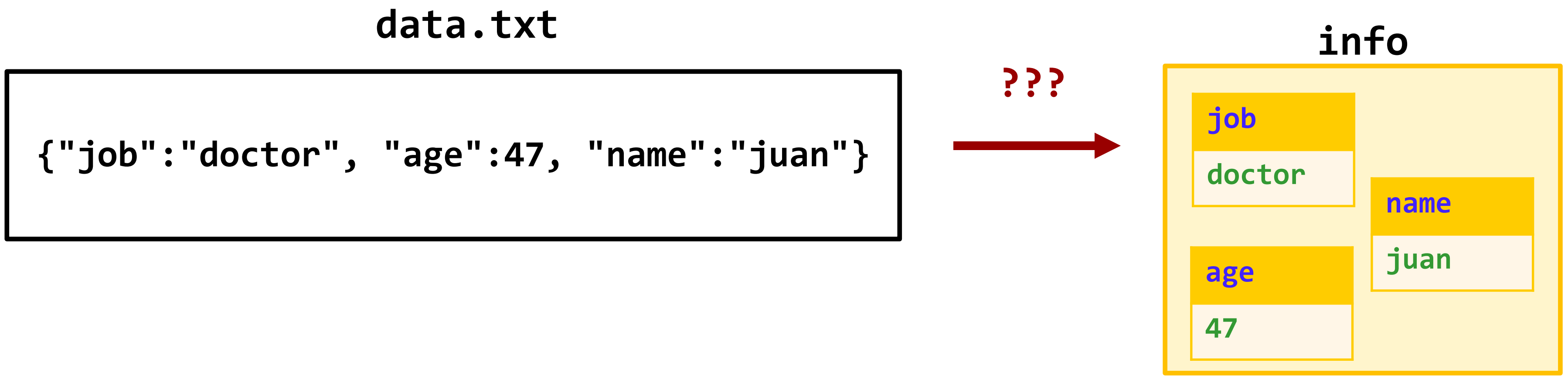
- Does this work?
  - Yes

Reading Complex Variables from Text Files

- How do we later read this text file to recreate the dictionary variable?

Reading Complex Variables from Text Files

- How do we later read this text file to recreate the dictionary variable?



- It would be challenging
  - Multiple times using `split()`

Moral of the Story

- Don't write entire lists or dictionaries (or complex variables) directly to a text file
- Let's try something else

Complex Data and Files

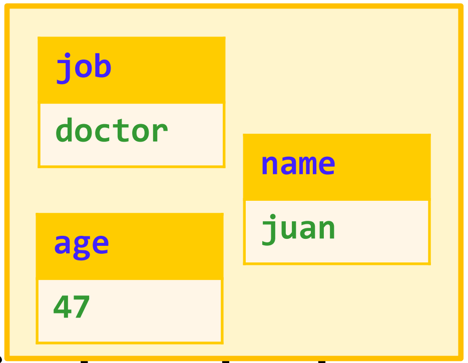
- How would you store the following list in a text file so you could restore it later?

0	1	2
doctor	juan	47

- CSV? Separate lines for each element?
  - This could work

Complex Data and Files

- How would you store the following dictionary in a text file so you could restore it later?



- This is more of a headache
  – Not particularly appealing

Recall: Kinds of Files

- Files are either stored as **Text** or **Binary**
- Text files store data in human-readable formats
  - Ex: Simple text files (.txt) or web pages (.html)
- Binary files data in computer-readable formats
  - Ex: pictures (.jpg), music (.mp3), or Word doc (.docx)

Binary Files

- We can store complex data directly in computer-readable format
  - This will **not be** human readable

- Data can be directly read back from file into variable

- Python calls this **Pickling** ("to preserve")
  - Sometimes called "serialization" or "marshaling"

Binary File Access Modes

Mode	Description
`"rb"`	**Read** from a file. If the file doesn't exist, Python will generate an error.
`"wb"`	**Write** to a file. If the file exists, its contents are overwritten. If the file doesn't exist, it's created.
`"ab"`	**Append** a file. If the file exists, new data is appended to it. If the file doesn't exist, it's created.

Pickling Data

- `pickle` module allows you to pickle and store more complex data in a file
- You can pickle a variety of objects including numbers, strings, tuples, lists, and dictionaries
- Pickling is pretty simple – write a pickled object to a file

Writing to a Binary File

Four Step Process

1. Import **pickle**
2. Open the file for **writing**
3. Write to the file
4. Close the file

Opening a Binary File for Writing

- Just like before, we use `open()`

```
fileOut = open("words.bin", "wb")
```

filename · *file access mode*

- Specify you want to **W**riting to a **B**inary file (output)

Writing To a Binary File

- `dump()` writes variable to file

```
pickle.dump(someVariable, fileOut)
```

variable

file

Writing To a Binary File

- Example

```
primary = ["red", "green", "blue"]
fileOut = open("colors.bin", "wb")
pickle.dump(primary, fileOut)
fileOut.close()
```

Reading from a Binary File

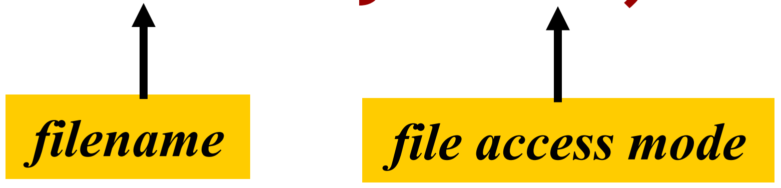
Four Step Process

1. Import **pickle**
2. Open the file for **reading**
3. Read from the file
4. Close the file

Opening a Binary File for Reading

- Just like before, we use `open()`

```
fileIn = open("words.bin", "rb")
```



- Specify you want to **R**ead from a **B**inary file (input)

Reading from a Binary File

- `load()` reads a variable from a binary file

```
someVariable = pickle.load(fileIn)
```

variable to store file data into

file

Reading from a Binary File

- Example

```
import pickle

fileIn = open("colors.bin", "rb")
primaryColors = pickle.load(fileIn)
fileIn.close()
```

What's the catch?

- You can `pickle` multiple variables to the same file

```
pickle.dump("hello", fileIn)
pickle.dump([4, 5, 2], fileIn)
```

- But <u>you</u> have to know how many variables are in the file when loading
  - There is no `for` loop trick to read all the variables in the file (like with text files)