[220 / 319] JSON

Department of Computer Sciences
University of Wisconsin-Madison

Readings:
Chapter 16 of Sweigart book

Worksheet practice with nesting

Learning Objectives

JSON:

- interpret data format
- differences with Python syntax
- deserialize data from JSON files to use in Python program (read)
- serialize data into JSON files for long term storage (write)

Read: Sweigart Ch 16
https://automatetheboringstuff.com/2e/chapter16/

“JSON and APIs” to the end

Python Data Structures and File Formats

Python

```
[
 [“name”, “x”, “y”],
 [“alice”, 100, 150],
 [“bob”, -10, 80]
]
```

list of lists

File

```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

We can use CSV files to store data we would want in lists of lists

Python Data Structures and File Formats

Python

```
[
 [“name”, “x”, “y”],
 [“alice”, 100, 150],
 [“bob”, -10, 80]
]
```

list of lists

File

```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

dict of dicts

?

Python Data Structures and File Formats

Python

```
[
 [“name”, “x”, “y”],
 [“alice”, 100, 150],
 [“bob”, -10, 80]
]
```

list of lists

File

```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

dict of dicts

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

JSON file

Python Data Structures and File Formats

JSON files look almost identical to Python code for data structures!

```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

dicts use curly braces

```
{
  "alice": {
    "age": 40,
    "scores": [10,20,19]},
  "bob": {
    "age": 45,
    "scores": [15,23,17,15]}
}
```

JSON file

Python Data Structures and File Formats

JSON files look almost identical to Python code for data structures!

```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

keys are separated from values with a colon

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

JSON file

Python Data Structures and File Formats

Python

File

JSON files look almost identical to Python code for data structures!

```
[
  [“name”, “x”, “y”],
  [“alice”, 100, 150],
  [“bob”, -10, 80]
]
```

list of lists

```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

lists use square brackets

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

dict of dicts

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

JSON file

Python Data Structures and File Formats

Python

File

JSON files look almost identical to Python code for data structures!

```
[
  [“name”, “x”, “y”],
  [“alice”, 100, 150],
  [“bob”, -10, 80]
]
```

list of lists

```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

strings are in quotes

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

dict of dicts

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

JSON file

Python Data Structures and File Formats

Python

File

JSON files look almost identical to Python code for data structures!

```
[
  [“name”, “x”, “y”],
  [“alice”, 100, 150],
  [“bob”, -10, 80]
]
```

list of lists

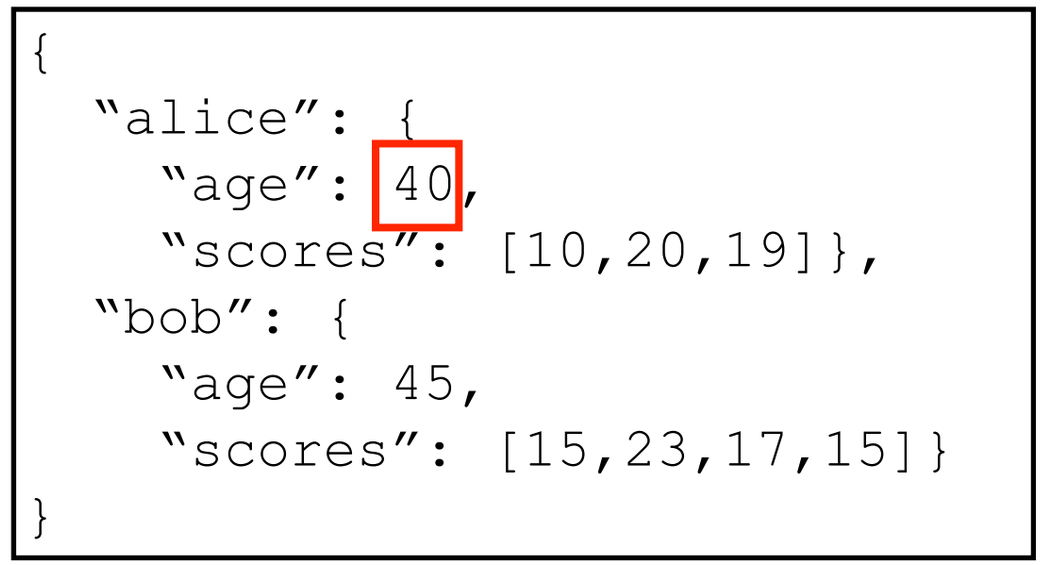
```
name,x,y
alice,100,150
bob,-10,80
```

CSV file

integers look like integers

```
{
  “alice”: {
    “age”: 40,
    “scores”: [10,20,19]},
  “bob”: {
    “age”: 45,
    “scores”: [15,23,17,15]}
}
```

dict of dicts



JSON file

JSON

Stands for **JavaScript Object Notation**

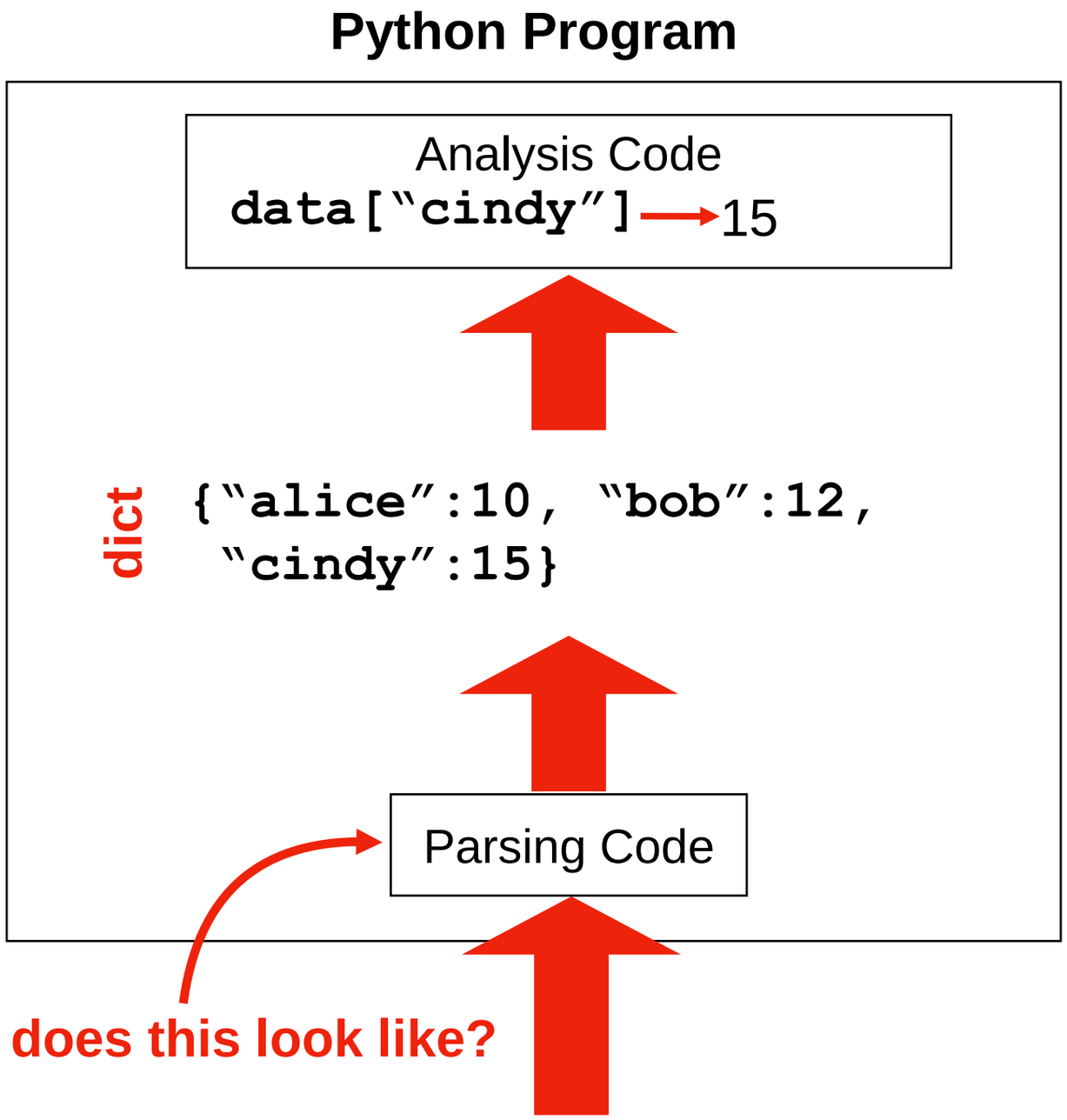
- JavaScript is a language for web development
- JSON was developed for JavaScript programs to store/share data
- JSON looks like Python code because JavaScript is similar to Python

Minor JavaScript vs. Python differences:

	Python	JSON
Booleans	True, False	true, false
No value	None	null
Quotes	Single (‘) or double (“)	Only double (“)
Commas	Extra allowed: [1,2,]	No extra: [1,2]
Keys	Any type: {3: “three”}	Str only: {“3”: “three”}

remember these!

Reading JSON Files



JSON file saved somewhere

```
{
  “alice”: 10,
  “bob”: 12,
  “cindy”: 15
}
```

Reading JSON Files

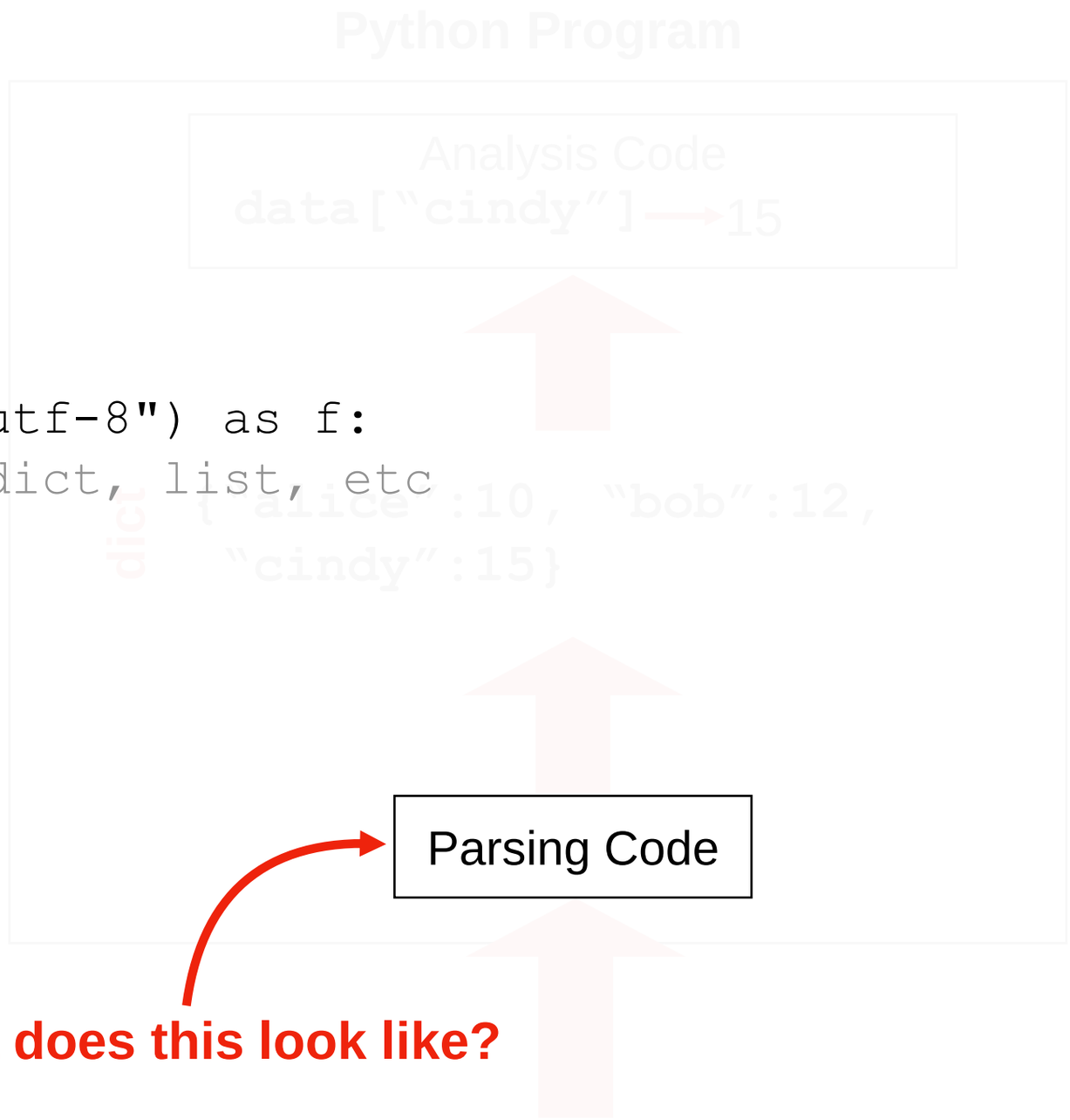
```
import json

def read_json(path):
    with open(path, encoding="utf-8") as f:
        return json.load(f) # dict, list, etc
```

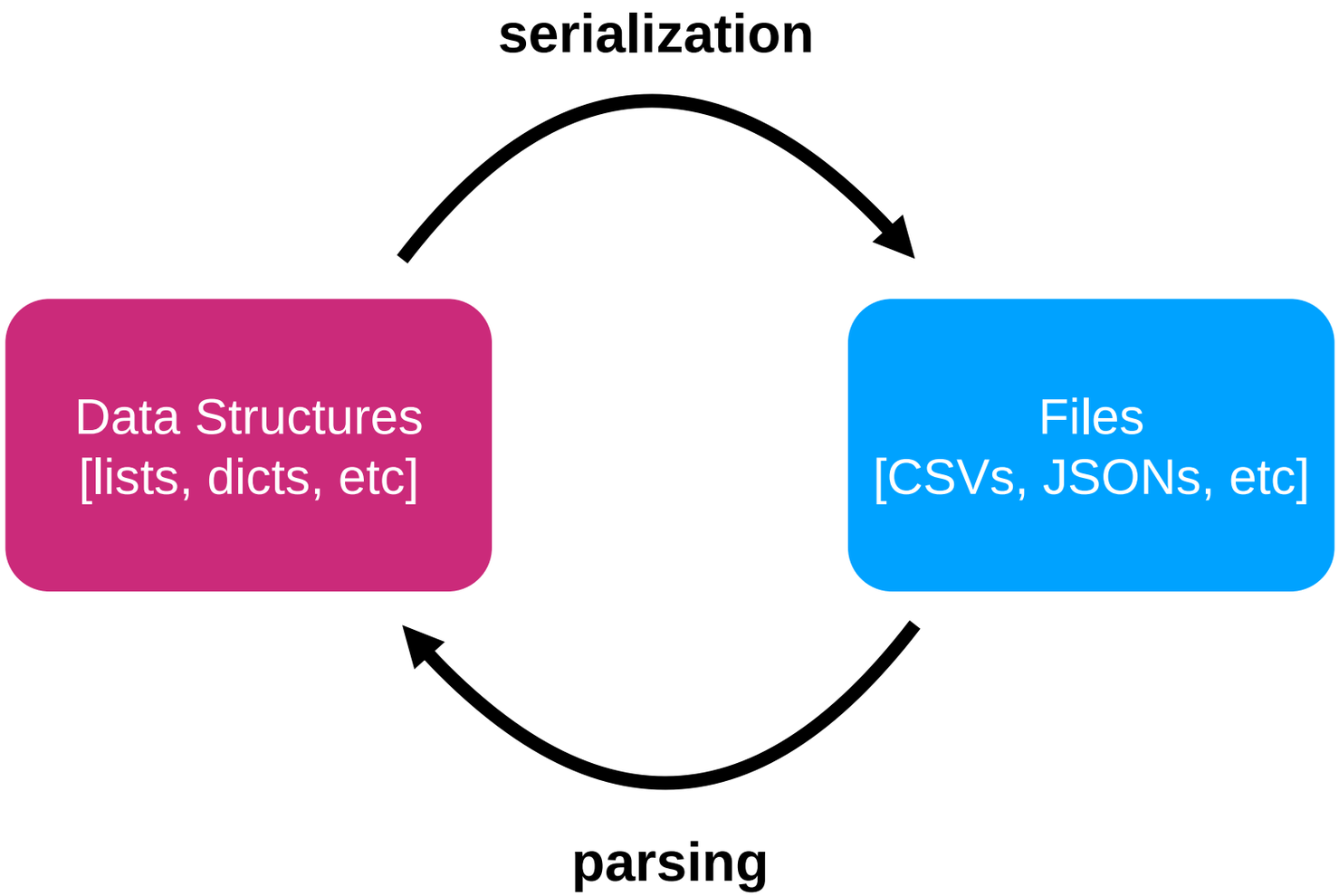
CTRL + C

don't need to understand this snippet yet

what about writing new files?



Data Structures and Files



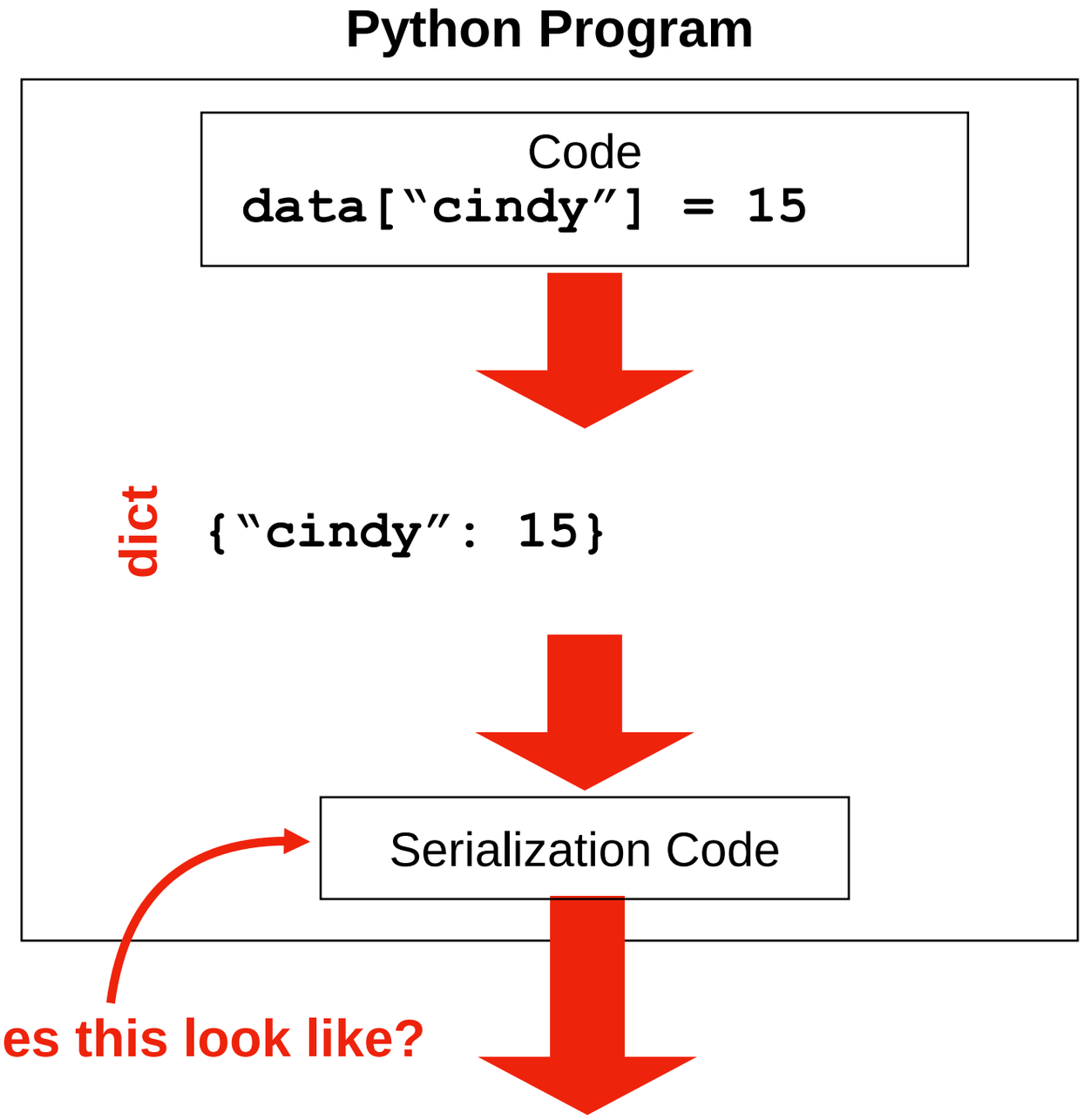
why not just have data structures?

because our data needs to live somewhere when our programs aren't running

why not just have files?

slow, and Python doesn't understand structure until it is parsed

Writing JSON Files



JSON file saved somewhere

```
{
  "cindy": 15
}
```

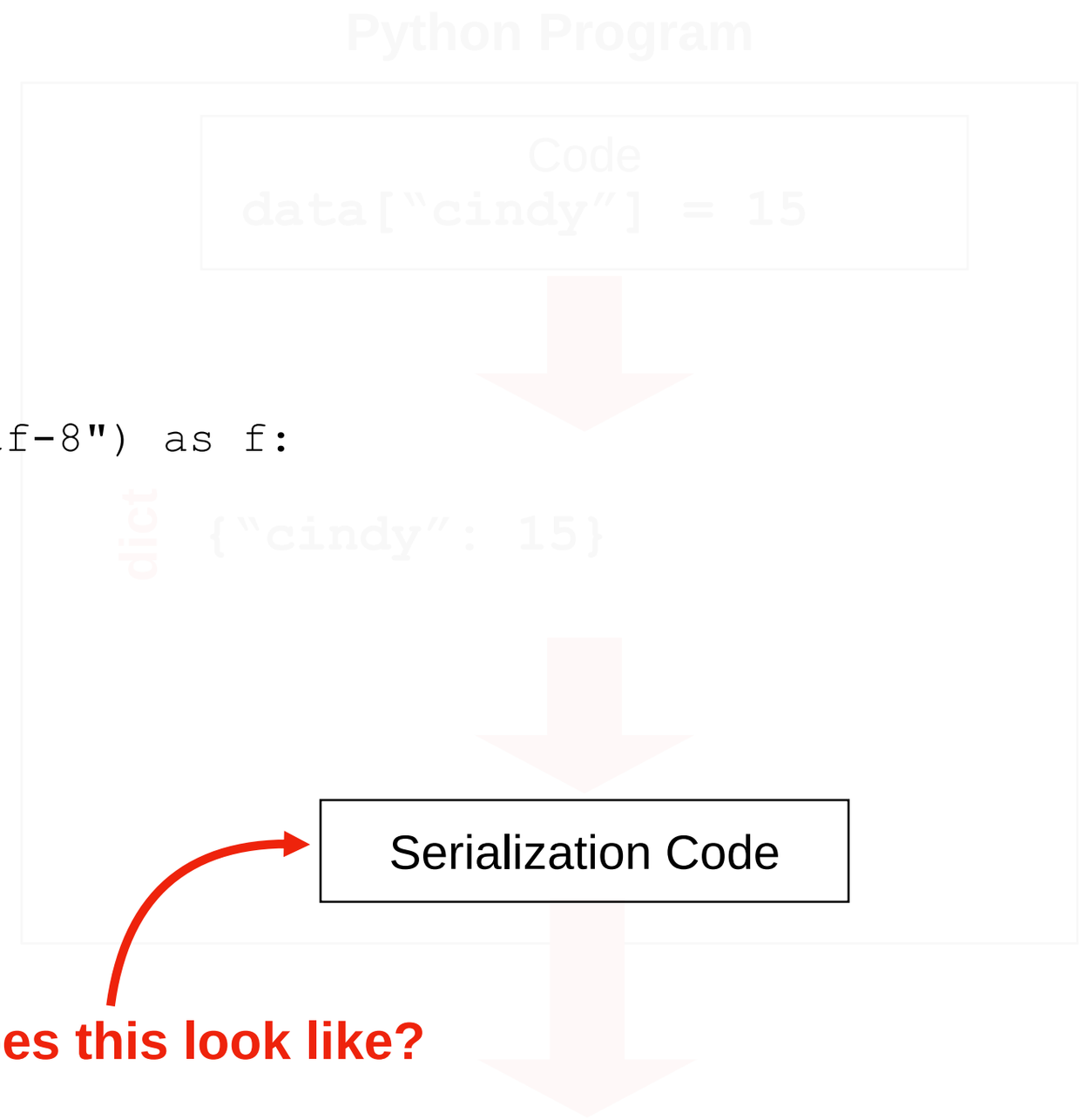
Writing JSON Files

```
import json

# data is a dict, list, etc
def write_json(path, data):
    with open(path, 'w', encoding="utf-8") as f:
        json.dump(data, f, indent=2)
```

CTRL + C

don't need to understand this snippet yet



Example: Sum of numbers (simple JSON)

Goal: count the numbers in a list saved as a JSON file

Input:

- Location of the JSON file

Output:

- The sum

Example:

fileA.json

```
[1,2,3]
```

output 6

Example: Score Tracker

Goal: record scores (save across runs) and print average

Input:

- A **name** and a **score** to record

Output:

- Running average for that person

Example:

"Enter player name and score": **alice 10**
Alice Avg: 10
"Enter player name and score": **alice 20**
Alice Avg: 15
"Enter player name and score": **bob 13**
Bob Avg: 13

Example – Exploring kiva.json

Goal: explore a real-world JSON file

kiva.json


```
{
  "data": {
    "lend": {
      "loans": {
        "values": [
          {
            "name": "Polikseni",
            "description": "Polikseni is 70 years old and married. She and her husband are both retired and their main income is a retirement pension of $106 a month for Polikseni and disability income for her husband of $289 a month. <br /><br />Polikseni's husband, even though disabled, works in a very small shop as a watchmaker on short hours, just to provide additional income for his family and to feel useful. Polikseni's husband needs constant medical treatment due to his health problems. She requested another loan, which she will use to continue paying for the therapy her husband needs. With a part of the loan, she is going to pay the remainder of the previous loan.",
            "loanAmount": "1325.00",
            "geocode": {
              "city": "Korce",
              "country": {
                "name": "Albania",
                "region": "Eastern Europe",
                "fundsLentInCountry": 9051250
              }
            }
          }, …
}
```

Challenge - Demo 4: Prime Cache

Goal: find number of primes less than N, cache previous return vals

Input:

- An integer N

Output:

- How many primes are less than that number

Challenge - Demo 5: Upper Autocomplete

Goal: record scores (save across runs) and print average

Input:

- A complete phrase
- A partial phrase ending with a *

Output:

- The upper case version of it
- Options to autocomplete

autocomplete must work
across multiple runs

Example:

msg: hi
HI
msg: hello
HELLO
msg: h*
1: hi
2: hello
select: 1
HI