Week 3: Fundamentals of coding with python

SESSION 5

RECAP

Topics

-Basic Syntax
-Data Types(string,integer,boolean)
-Compound data structures(list,set,tuple,dictionary)
-Control Flows(If, elif, for loops, **while loops**)
-Python built-in functions and methods
-Error handling
-Functions
-List and dictionary comprehension
-Debugging your python code

List comprehension

Allows you to achieve more with less

List comprehension

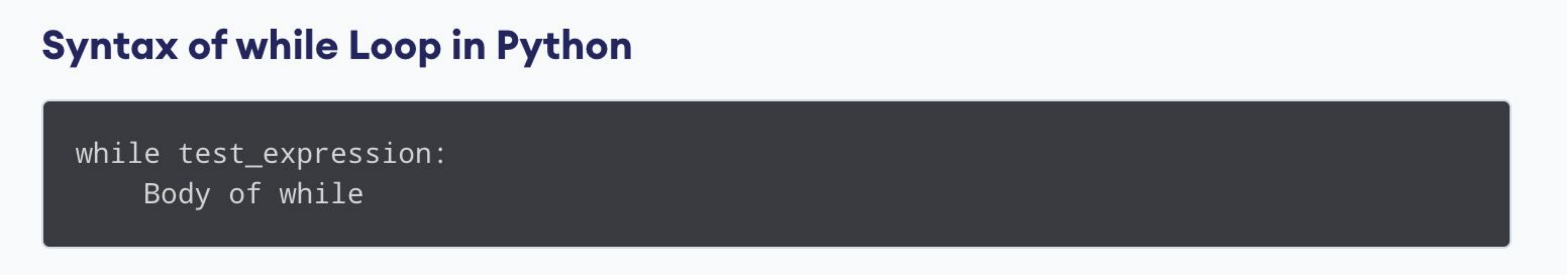
list_variable = [*expression* for *item* in *iterable* if *condition* == True]

Dictionary comprehension

Dictionary comprehension

```
dict_variable = {key:value for (key,value) in dictonary.items()}
```

WHILE LOOP



Syntax of while Loop in Python

```
while test_expression:
    Body of while
```

continue and
break statements

PROBLEM SOLVING

ALGORITHMS

If problem solving is a central part of computer science, then the solutions that you create through the problem solving process are also important. In computer science, we refer to these solutions as **algorithms.** An algorithm is a step by step list of instructions that if followed exactly will solve the problem under consideration.

EXERCISES 1

Create a function that takes two arguments:
the original price and the discount percentage as integers and returns the final price after the discount.

dis(1500, 50) ➞ 750

dis(89, 20) ➞ 71.2

dis(100, 75) ➞ 25

EXERCISES 2

Given the radius of a circle and the area of a square, return True if the circumference of the circle is greater than the square's perimeter and False if the square's perimeter is greater than the circumference of the circle.

Notes

You can use Pi to 2 decimal places (3.14).

Circumference of a circle equals 2 * Pi * radius.

To find the perimeter of a square using its area, find the square root of area (to get side length) and multiply that by 4.

EXERCISES 3

Create a function that takes a string and returns the number (count) of vowels contained within it.

Examples

count_vowels("Celebration") → 5

count_vowels("Palm") → 1

count_vowels("Prediction") → 4

a, e, i, o, u are considered vowels.

EXERCISES 5

Create a function that takes three arguments a, b, c and returns the sum of the numbers that are evenly divided by c from the range a, b inclusive.

evenly_divisible(1, 10, 20) → 0
# No number between 1 and 10 can be evenly divided by 20.

evenly_divisible(1, 10, 2) → 30
# 2 + 4 + 6 + 8 + 10 = 30

evenly_divisible(1, 10, 3) → 18
# 3 + 6 + 9 = 18

EXERCISES 5

Write a program that creates a personalised invite for your friends to your birthday party.
Your program should include your friend’s name, a different pass code for each friend and the invite message.
NB: You can use a dictionary to store your friends name and their pass code.

How are you feeling?

RED

I have no idea what you're talking about

YELLOW

I have some questions but feel like I understand some things

GREEN

I feel comfortable with everything you've said

SESSION 6

DEBUGGING

Programming is a complex process. Since it is done by human beings, errors may often occur. Programming errors are called **bugs** and the process of tracking them down and correcting them is called **debugging**. Some claim that in 1945, a dead moth caused a problem on relay number 70, panel F, of one of the first computers at Harvard, and the term **bug** has remained in use since. For more about this historic event, see first bug.

Three kinds of errors can occur in a program: syntax errors, runtime errors, and semantic errors. It is useful to distinguish between them in order to track them down more quickly.

SYNTAX

Python can only execute a program if the program is syntactically correct; otherwise, the process fails and returns an error message. **Syntax** refers to the structure of a program and the rules about that structure. For example, in English, a sentence must begin with a capital letter and end with a period. this sentence contains a **syntax error**. So does this one

RUNTIME

The second type of error is a runtime error, so called because the error does not appear until you run the program. These errors are also called **exceptions** because they usually indicate that something exceptional (and bad) has happened.

Runtime errors are rare in the simple programs you will see in the first few chapters, so it might be awhile before you encounter one.

SEMANTICS

The third type of error is the **semantic error**. If there is a semantic error in your program, it will run successfully in the sense that the computer will not generate any error messages. However, your program will not do the right thing. It will do something else. Specifically, it will do what you told it to do.

The problem is that the program you wrote is not the program you wanted to write. The meaning of the program (its semantics) is wrong. Identifying semantic errors can be tricky because it requires you to work backward by looking at the output of the program and trying to figure out what it is doing.

ERROR TYPES

-ParseError

-TypeError

-NameError

-ValueError

Parse Error/Syntax Error

Parse errors happen when you make an error in the syntax of your program. Syntax errors are like making grammatical errors in writing. If you don’t use periods and commas in your writing then you are making it hard for other readers to figure out what you are trying to say. Similarly Python has certain grammatical rules that must be followed or else Python can’t figure out what you are trying to say.

Usually ParseErrors can be traced back to missing punctuation characters, such as parentheses, quotation marks, or commas. Remember that in Python commas are used to separate parameters to functions. Paretheses must be balanced, or else Python thinks that you are trying to include everything that follows as a parameter to some function.

TypeError

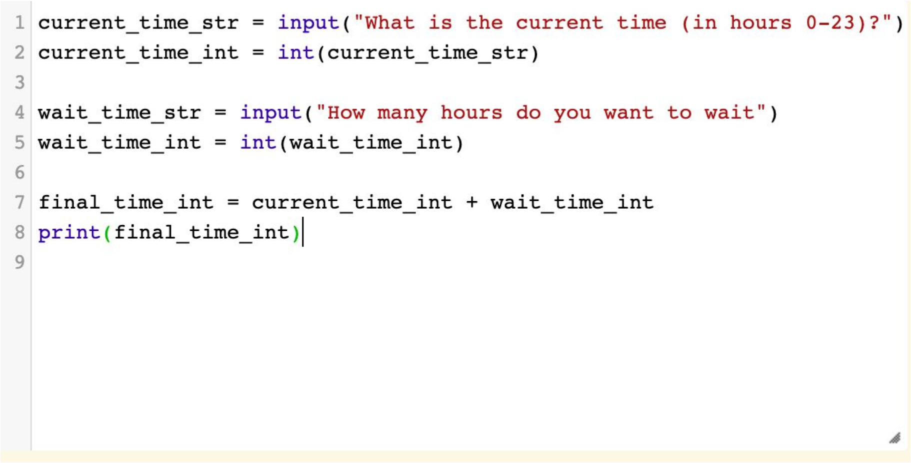
TypeErrors occur when you you try to combine two objects that are not compatible. For example you try to add together an integer and a string. Usually type errors can be isolated to lines that are using mathematical operators, and usually the line number given by the error message is an accurate indication of the line.

NameError

Name errors almost always mean that you have used a variable before it has a value. Often NameErrors are simply caused by typos in your code. They can be hard to spot if you don't have a good eye for catching spelling mistakes. Other times you may simply mis-remember the name of a variable or even a function you want to call. You have seen one example of a NameError at the beginning of this section.

ValueError

Value errors occur when you pass a parameter to a function and the function is expecting a certain limitations on the values, and the value passed is not compatible. (Run code in note)



```
current_time_str = input("What is the current time (in hours 0-23)?")
current_time_int = int(current_time_str)

wait_time_str = input("How many hours do you want to wait")
wait_time_int = int(wait_time_int)

final_time_int = current_time_int + wait_time_int
print(final_time_int)

```

How are you feeling?

RED

I have no idea what you’re talking about

YELLOW

I have some questions but feel like I understand some things

GREEN

I feel comfortable with everything you’ve said

EXERCISES 7

Fix the code in the code tab to pass this challenge (only syntax errors). Look at the examples below to get an idea of what the function should do.

Examples
cubes(3) → 27
cubes(5) → 125
cubes(10) → 1000
Notes
READ EVERY WORD CAREFULLY, CHARACTER BY CHARACTER!
Don't overthink this challenge; it's not supposed to be hard.

<code>

```
def cube():
    retrun x ***3

#çalling function to test our code
cubes(3)
```

EXERCISES 8

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_int)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

EXERCISES 9

A student learning Python was trying to make a function. His code should concatenate a passed string name with string "BlackDisrupt" and store it in a variable called result. He needs your help to fix this code.

Examples
name_string("Mubashir") → "MubashirBlackDisrupt"
name_string("Matt") → "MattBlackDisrupt"
name_string("python") → "pythonBlackDisrupt"
Notes
Don't forget to return the result.

<code>

```
def name_string(name):
        b == "BlackDisrupt"
        result == name + b
        print result

#çalling function to test our code
name_string("Mubashir")
```

EXERCISES 10

Help fix all the bugs in the function increment_items! It is intended to add 1 to every element in the list!

Examples
increment_items([0, 1, 2, 3]) → [1, 2, 3, 4]
increment_items([2, 4, 6, 8]) → [3, 5, 7, 9]
increment_items([-1, -2, -3, -4]) → [0, -1, -2, -3]
Notes
Make sure to read every line carefully.

```
<code>
def increment_items(lst)
    for i in list:
        i += 1
    return list
```

How are you feeling?

RED

I have no idea what you're talking about

YELLOW

I have some questions but feel like I understand some things

GREEN

I feel comfortable with everything you've said

Next Topics

-Python class(Object oriented programming, inheritance)
-Working with virtual environments and requirements files
-Python standard libraries/modules
    -request, math,os, glob, argparse, datetime, random
    -use dir() function to see the methods available in a module
-Introduction to API

INTRO TO OOP

WHAT IS OOP?

Object oriented programming(OOP) is a concept in programming where everything is seem as an object.

Fetching Data from API

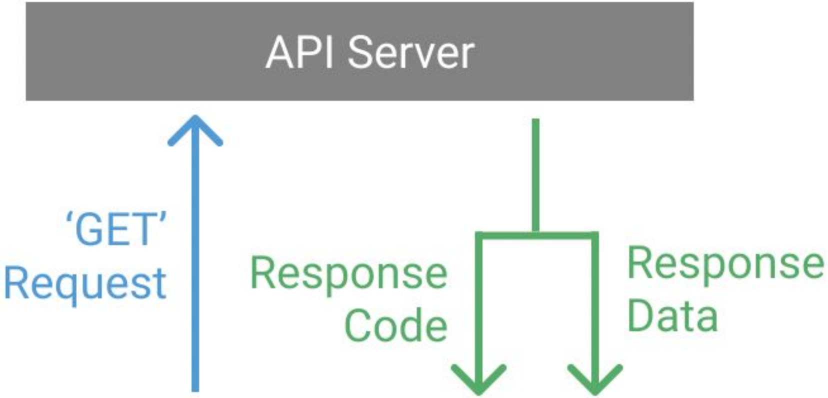
Fetching data from an API

…

What is an API?

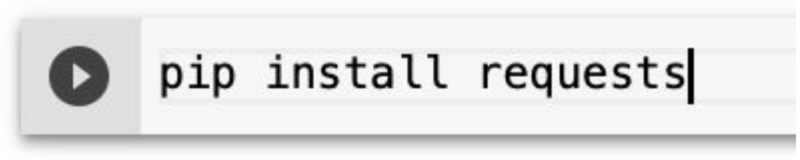
What is An API?

- An API, or Application Programming Interface, is a server that you can use to retrieve and send data to using code. APIs are most commonly used to retrieve data, and that will be the focus of this lesson.



What is an API?

- In order to work with APIs in Python, we need tools that will make those requests. In Python, the most common library for making requests and working with APIs is the requests library. The requests library isn't part of the standard Python library, so you'll need to install it to get started.



```
pip install requests
```

What is an API?

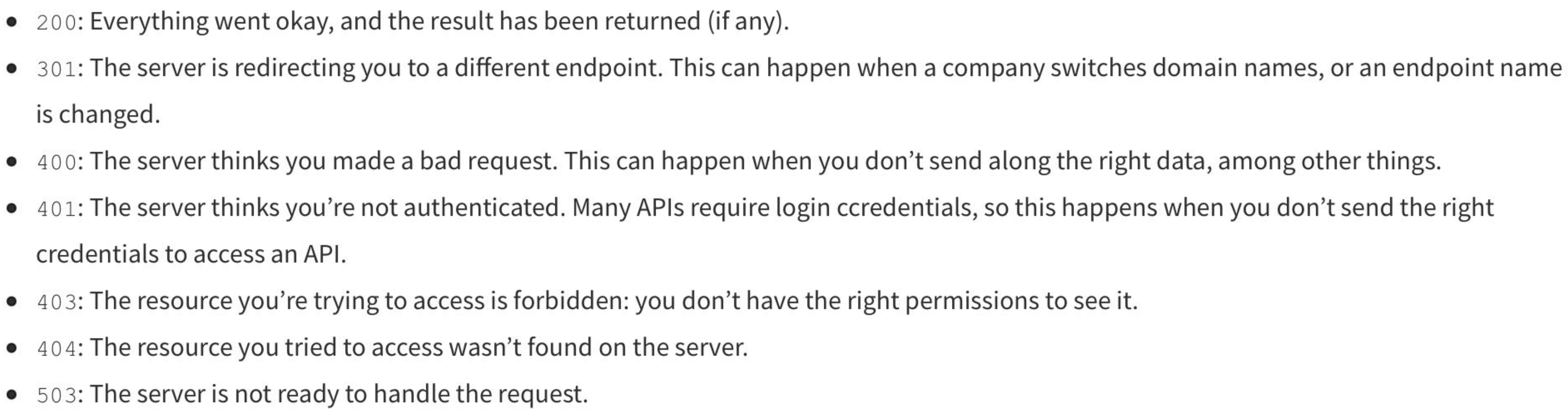
- There are many different types of requests. The most commonly used one, a GET request, is used to retrieve data. Because we'll just be working with retrieving data, our focus will be on making 'get' requests.

- When we make a request, the response from the API comes with a response code which tells us whether our request was successful. Response codes are important because they immediately tell us if something went wrong.

- To make a 'GET' request, we'll use the `requests.get()` function, which requires one argument — the URL we want to make the request to. We'll start by making a request to an API endpoint that doesn't exist, so we can see what that response code looks like.

API Status Codes:

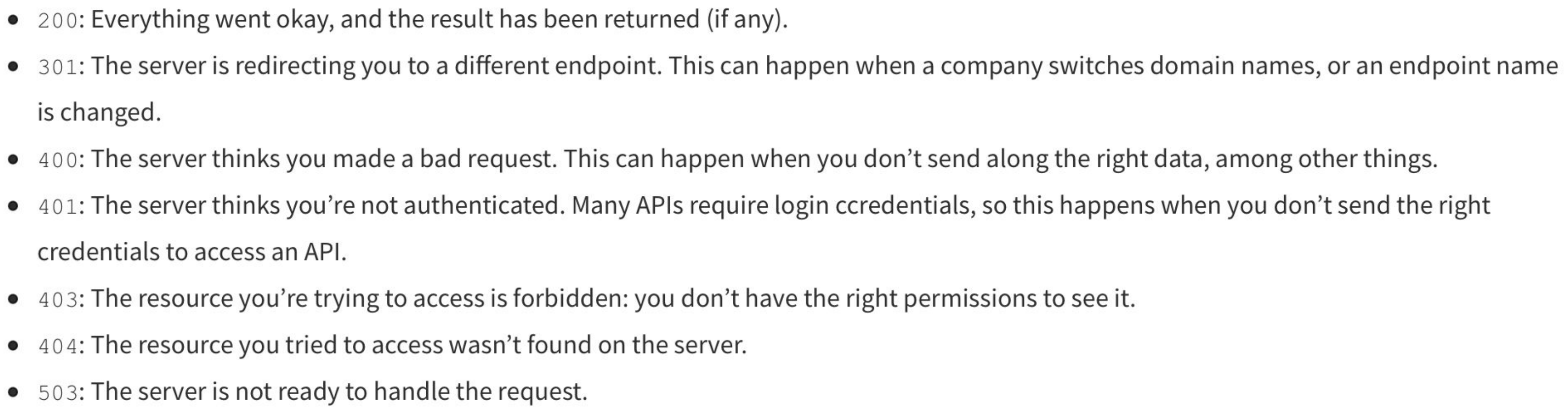
- **Status codes are returned with every request that is made to a web server. Status codes indicate information about what happened with a request. Here are some codes that are relevant to *GET* requests:**



- `200`: Everything went okay, and the result has been returned (if any).
- `301`: The server is redirecting you to a different endpoint. This can happen when a company switches domain names, or an endpoint name is changed.
- `400`: The server thinks you made a bad request. This can happen when you don’t send along the right data, among other things.
- `401`: The server thinks you’re not authenticated. Many APIs require login ccredentials, so this happens when you don’t send the right credentials to access an API.
- `403`: The resource you’re trying to access is forbidden: you don’t have the right permissions to see it.
- `404`: The resource you tried to access wasn’t found on the server.
- `503`: The server is not ready to handle the request.

API Status Codes:

- You might notice that all of the status codes that begin with a ‘4’ indicate some sort of error. The first number of status codes indicate their categorization. This is useful — you can know that if your status code starts with a ‘2’ it was successful and if it starts with a ‘4’ or ‘5’ there was an error.



- `200`: Everything went okay, and the result has been returned (if any).
- `301`: The server is redirecting you to a different endpoint. This can happen when a company switches domain names, or an endpoint name is changed.
- `400`: The server thinks you made a bad request. This can happen when you don’t send along the right data, among other things.
- `401`: The server thinks you’re not authenticated. Many APIs require login ccredentials, so this happens when you don’t send the right credentials to access an API.
- `403`: The resource you’re trying to access is forbidden: you don’t have the right permissions to see it.
- `404`: The resource you tried to access wasn’t found on the server.
- `503`: The server is not ready to handle the request.

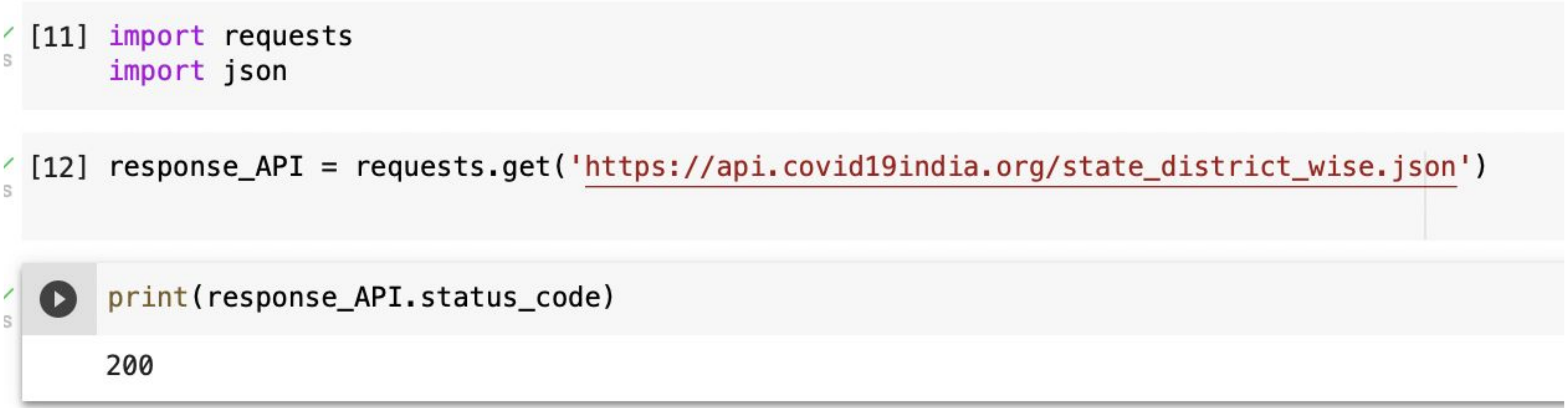
What is an API?

- We will use the COVID19-India API to fetch data of the cases from the state-wise list. It’s a great API for learning because it has a very simple design, and doesn’t require authentication. You will later learn how to use an API that requires authentication.

- Often there will be multiple APIs available on a particular server. Each of these APIs are commonly called endpoints.

What is an API?

- Let us start by making a GET request to the endpoint using the requests library:



```
[11] import requests
     import json
```

```
[12] response_API = requests.get('https://api.covid19india.org/state_district_wise.json')
```

```
print(response_API.status_code)
```

```
200
```

How are you feeling?

RED

I have no idea what you're talking about

YELLOW

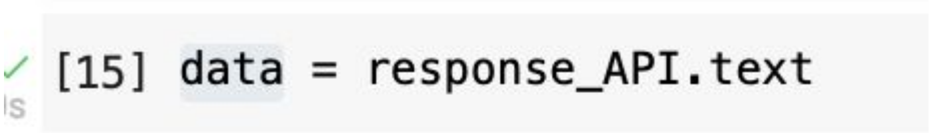
I have some questions but feel like I understand some things

GREEN

I feel comfortable with everything you've said

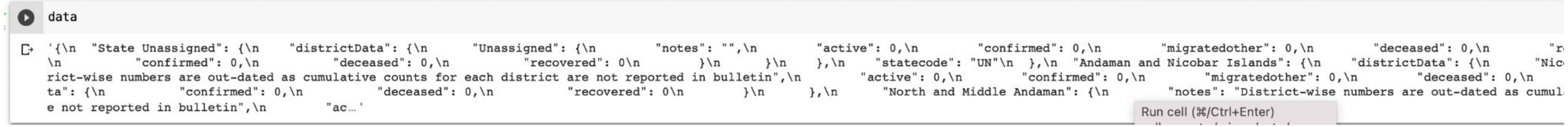
Get Data From API

- After making a healthy connection with the API, the next task is to pull the data from the API. Look at the below code!



```
[15] data = response_API.text
```

- What happens if we just print this ‘data’ variable? You will see some very confusing text. (Try it!) Let us format this data in a way that makes it readable.

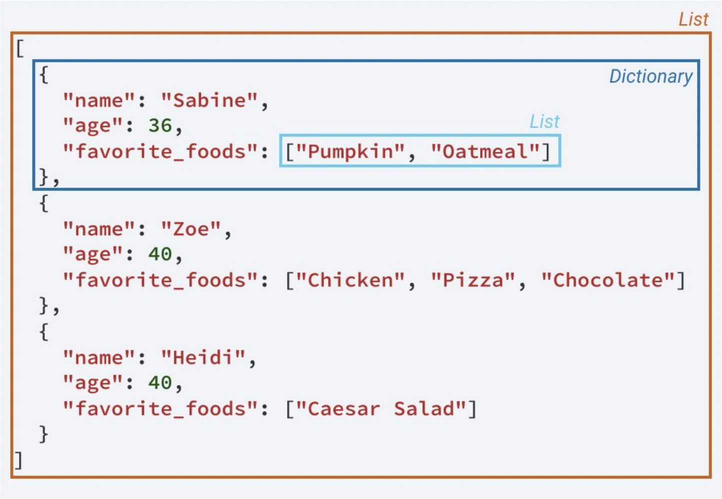


```
data
```

```
'{\n  "State Unassigned": {\n    "districtData": {\n      "Unassigned": {\n        "notes": "",\n        "active": 0,\n        "confirmed": 0,\n        "migratedother": 0,\n        "deceased": 0,\n        "r
\n          "confirmed": 0,\n          "deceased": 0,\n          "recovered": 0\n        }\n      }\n    },\n    "statecode": "UN"\n  },\n  "Andaman and Nicobar Islands": {\n    "districtData": {\n      "Nic
rict-wise numbers are out-dated as cumulative counts for each district are not reported in bulletin",\n        "active": 0,\n        "confirmed": 0,\n        "migratedother": 0,\n        "deceased": 0,\n
ta": {\n          "confirmed": 0,\n          "deceased": 0,\n          "recovered": 0\n        }\n      },\n      "North and Middle Andaman": {\n        "notes": "District-wise numbers are out-dated as cumul
e not reported in bulletin",\n        "ac…'
```

JSON File Format

- JSON (JavaScript Object Notation) is the language of APIs. JSON is a way to encode data structures that ensures that they are easily readable. JSON is the primary format in which data is passed back and forth to APIs, and most API servers will send their responses in JSON format.



```
[
  {
    "name": "Sabine",
    "age": 36,
    "favorite_foods": ["Pumpkin", "Oatmeal"]
  },
  {
    "name": "Zoe",
    "age": 40,
    "favorite_foods": ["Chicken", "Pizza", "Chocolate"]
  },
  {
    "name": "Heidi",
    "age": 40,
    "favorite_foods": ["Caesar Salad"]
  }
]
```

Parsing Data into JSON

Having extracted the data, it is now the time to convert the data into proper JSON format

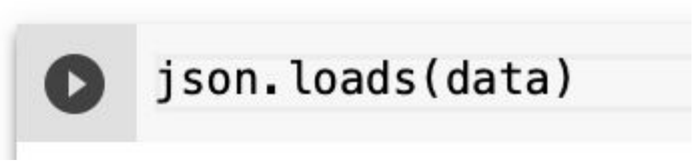
The json library has two main functions:

- json.dumps() — Takes in a Python object, and converts (dumps) it to a string.
- json.loads() — Takes a JSON string, and converts (loads) it to a Python object.

Parsing Data into JSON

The json.loads() function parses the data into a JSON format.

Run the following cell. What do you see?

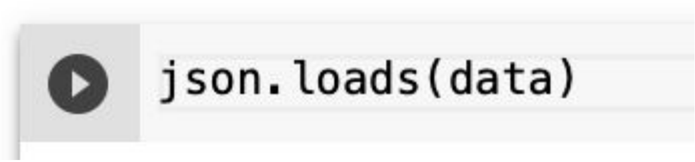


```
json.loads(data)
```

Parsing Data into JSON

The json.loads() function parses the data into a JSON format.

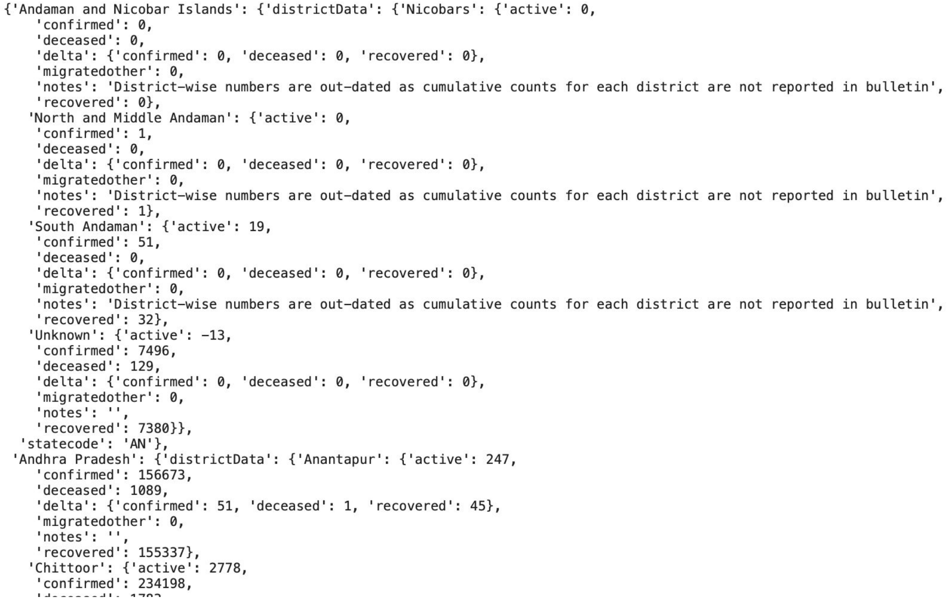
Run the following cell. What do you see?



```
json.loads(data)
```

Parsing Data into JSON

The JSON format contains data into a key-value format which resembles a Python dictionary. Can you tell what’s going on?



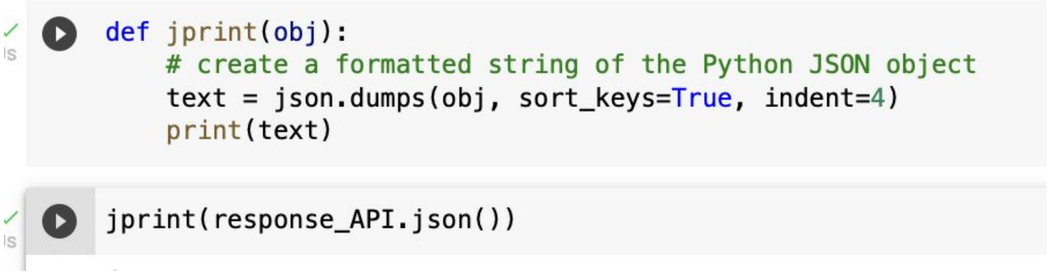
```
{'Andaman and Nicobar Islands': {'districtData': {'Nicobars': {'active': 0,
    'confirmed': 0,
    'deceased': 0,
    'delta': {'confirmed': 0, 'deceased': 0, 'recovered': 0},
    'migratedother': 0,
    'notes': 'District-wise numbers are out-dated as cumulative counts for each district are not reported in bulletin',
    'recovered': 0},
   'North and Middle Andaman': {'active': 0,
    'confirmed': 1,
    'deceased': 0,
    'delta': {'confirmed': 0, 'deceased': 0, 'recovered': 0},
    'migratedother': 0,
    'notes': 'District-wise numbers are out-dated as cumulative counts for each district are not reported in bulletin',
    'recovered': 1},
   'South Andaman': {'active': 19,
    'confirmed': 51,
    'deceased': 0,
    'delta': {'confirmed': 0, 'deceased': 0, 'recovered': 0},
    'migratedother': 0,
    'notes': 'District-wise numbers are out-dated as cumulative counts for each district are not reported in bulletin',
    'recovered': 32},
   'Unknown': {'active': -13,
    'confirmed': 7496,
    'deceased': 129,
    'delta': {'confirmed': 0, 'deceased': 0, 'recovered': 0},
    'migratedother': 0,
    'notes': '',
    'recovered': 7380}},
  'statecode': 'AN'},
 'Andhra Pradesh': {'districtData': {'Anantapur': {'active': 247,
    'confirmed': 156673,
    'deceased': 1089,
    'delta': {'confirmed': 51, 'deceased': 1, 'recovered': 45},
    'migratedother': 0,
    'notes': '',
    'recovered': 155337},
   'Chittoor': {'active': 2778,
    'confirmed': 234198,
    'deceased': 1782
```

Formatting JSON

It is a little difficult to understand the JSON data as seen in the previous slide. Let us format it using the following function, so that it is a bit clearer

```
def jprint(obj):
    text = json.dumps(obj,sort_keys=True, indent=4)
      print(text)
```

Use this function to format and print your JSON data.

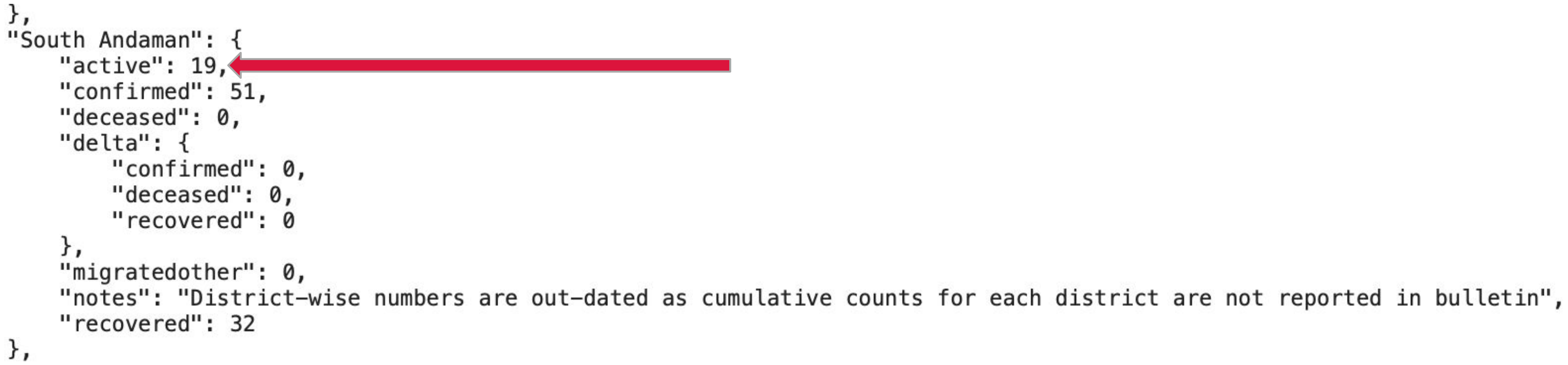


```
def jprint(obj):
    # create a formatted string of the Python JSON object
    text = json.dumps(obj, sort_keys=True, indent=4)
    print(text)
```

```
jprint(response_API.json())
```

See the difference!

From the JSON data we see that the number of ‘active’ cases in South Andaman is 19

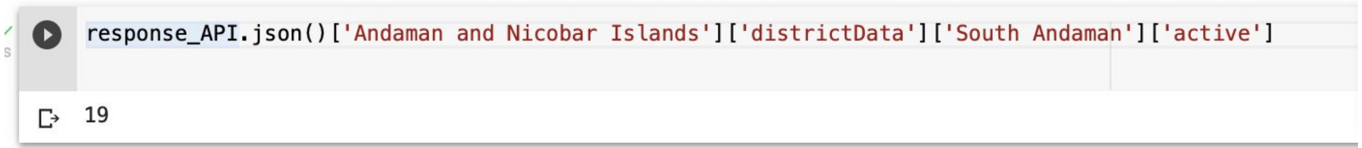


```
},
"South Andaman": {
    "active": 19,
    "confirmed": 51,
    "deceased": 0,
    "delta": {
        "confirmed": 0,
        "deceased": 0,
        "recovered": 0
    },
    "migratedother": 0,
    "notes": "District-wise numbers are out-dated as cumulative counts for each district are not reported in bulletin",
    "recovered": 32
},
```

But how to we get just this value? Without printing everything out ??

Accessing data

We can pull out and print the data using the key values as shown below.



```
response_API.json()['Andaman and Nicobar Islands']['districtData']['South Andaman']['active']
```

```
19
```

When we run the cell, you can see that we get the value 19.

Can you get the value for ‘deceased’ in ‘Andhra Pradesh’?

EXERCISE

The Pythagorean Theorem tells us that the length of the hypotenuse of a right triangle is related to the lengths of the other two sides. Look through the math module and see if you can find a function that will compute this relationship for you. Once you find it, write a short program to try it out.

EXERCISE

The Pythagorean Theorem tells us that the length of the hypotenuse of a right triangle is related to the lengths of the other two sides. Look through the math module and see if you can find a function that will compute this relationship for you. Once you find it, write a short program to try it out.

Thank You