API and Python training

Session 4

This session agenda

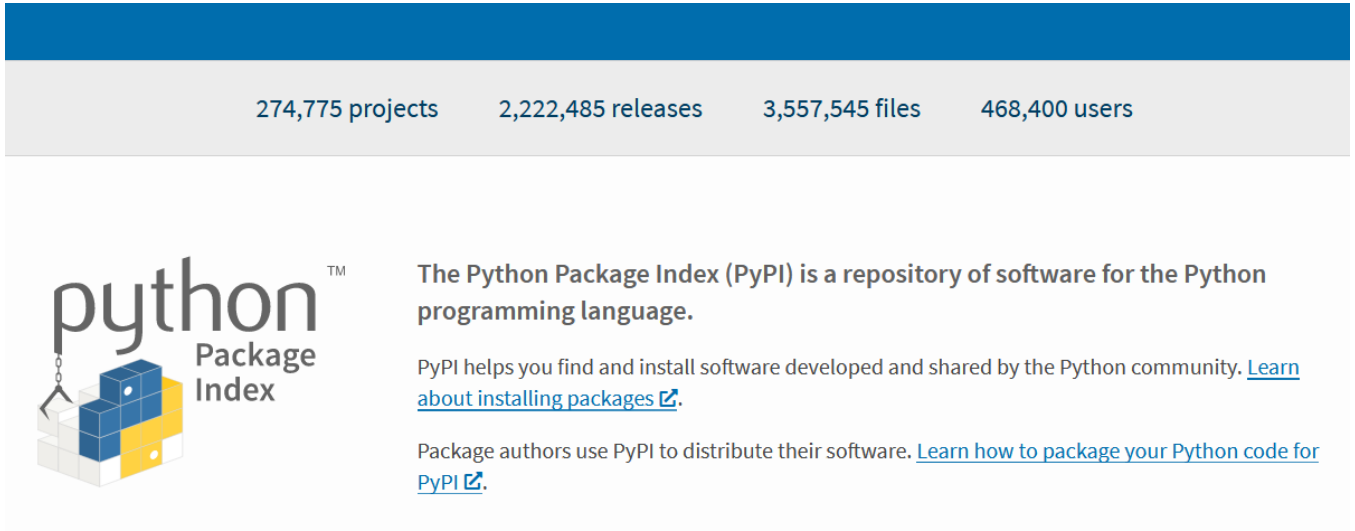
- Import packages
- Python *requests* library
- Making API requests
- Practice
- JSON
- Recap from last session – nested data structures
- Accessing JSON data
- Practice

Installing and importing modules

- Python module is just a normal Python file with .py
- You can split your code into **modules** to make it easier to understand and maintain
- **Packages** are set of modules, like folders
- Modules can be written on other languages and can be publically available, often called **libraries**
- Packages can use other packages, called **dependencies**
- Some packages already comes with Python, they are called **standard libraries**, you don't have to install them
- Downloaded from package repository https://pypi.org/ or its mirrors using **pip** utility, next slide

- Once the package is installed

use **import** statement **in the code**



https://stackoverflow.com/questions/19198166/whats-the-difference-between-a-module-and-a-library-in-python

https://realpython.com/python-import/#basic-python-import

Installing packages with PIP

- **Pip** comes with python by default – package installed (similar to npm in NodeJS)
- If you work in a terminal session use **pip install <package-name>** - this will download and install package, so you can import it in Python code
- Pip not only installs requested package, but also its dependencies
- As you can recall from last sessions libraries are stored in **lib** folder in **venv**

Useful pip commands:

pip list – get current packages installed in your system or venv ---------->

```
(venv) C:\dev\test_pr1>pip list
Package    Version
---------- ---------
certifi    2020.11.8
chardet    3.0.4
idna       2.10
pip        20.2.4
requests   2.25.0
setuptools 50.3.2
urllib3    1.26.2
```

pip freeze > requirements.txt - get a 'snapshot' or currently installed packaged and save to a file, so you (or someone else) can import packages in bulk ---------->

requirements.txt

```
colorama==0.4.4
Flask==1.1.2
future==0.18.2
itsdangerous==1.1.0
Jinja2==2.11.2
netmiko==3.3.2
numpy==1.19.4
```

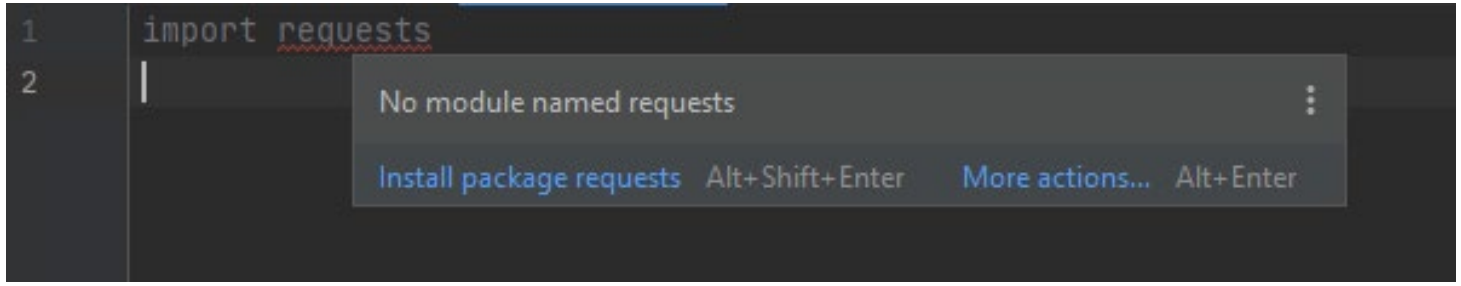
pip install -r requirements.txt - install all packages listed in the file --------->

```
(venv) C:\dev\new_project_01>pip install -r requirements.txt
Collecting colorama==0.4.3
  Using cached colorama-0.4.3-py2.py3-none-any.whl (15 kB)
Collecting Flask==1.1.2
  Using cached Flask-1.1.2-py2.py3-none-any.whl (94 kB)
Collecting future==0.18.2
  Using cached future-0.18.2.tar.gz (829 kB)
```

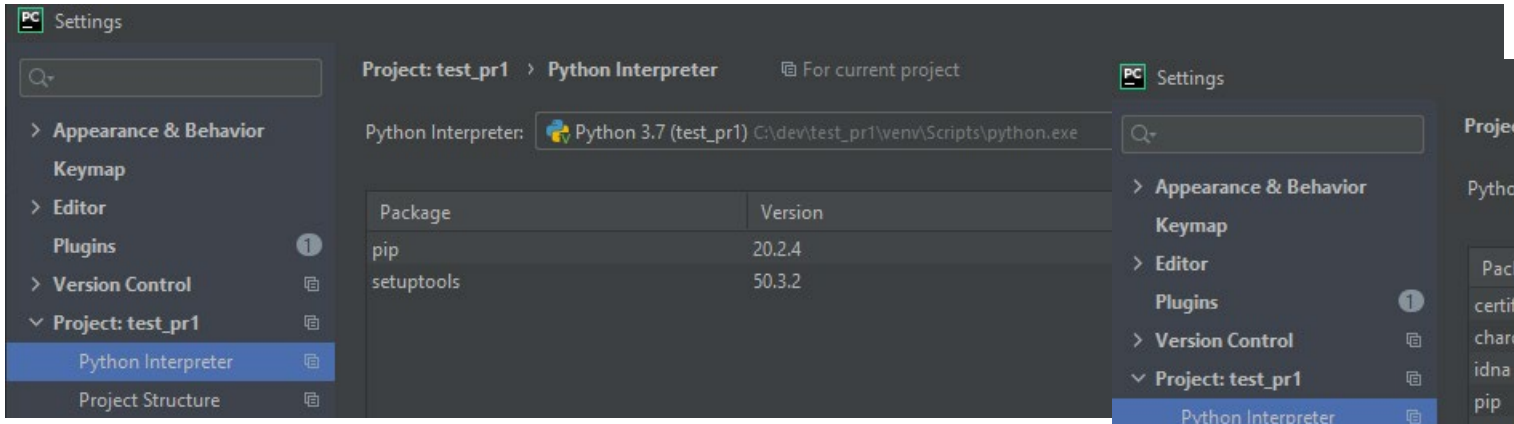
pip without any arguments shows available options (**help**)

Installing packages using IDE

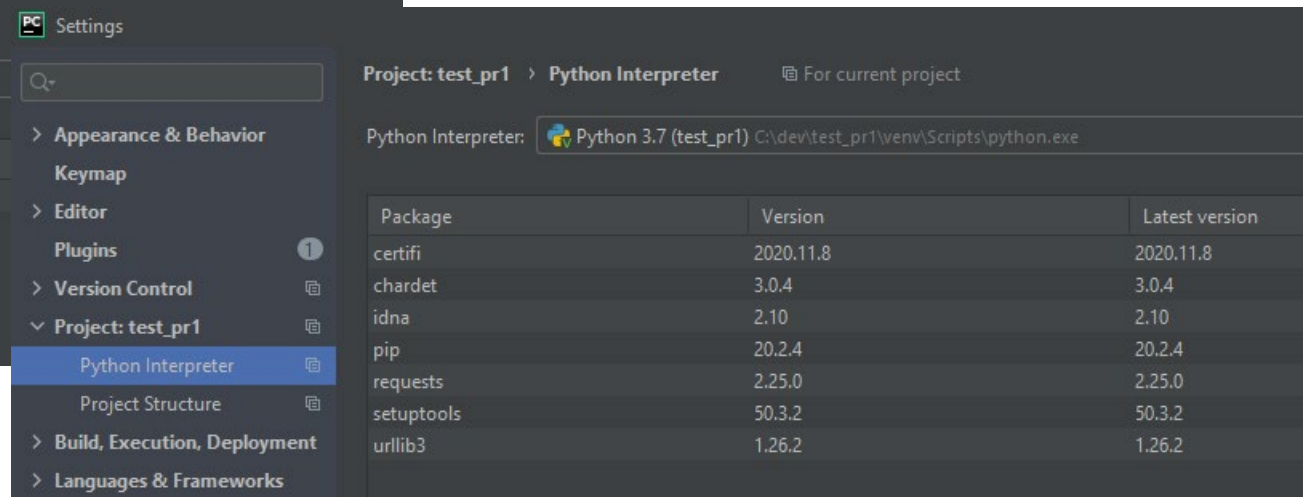
- IDE can recognize many packages
- Can install from the main window where you write code
- When you work in IDE and type module name it will ask you to install it
- Uses pip
- You can check and install packages manually in the project settings



Before (no dependencies):



After (note dependencies are installed):



Requests library

- The most popular library to make API calls with Python
- Vendors provide examples mostly using this library ---->
- Easy to use
- Offers synchronous (or blocking) API requests – there are also asynchronous libraries, can make multiple request at the same time - more scalable, but more complex - just FYI https://realpython.com/async-io-python/

- Excellent documentation https://requests.readthedocs.io
- Install it with **pip** or in IDE
- Use **import requests** in you code
- If you make requests to private systems which use self-signed or invalid certificates, you can disable certificate warning messages

```
# Added for using with sandbox, comment the line below for using in production
requests.packages.urllib3.disable_warnings()
```

Table API Python examples

Examples that demonstrate how to use the Table API with the Python language.

Example URLs

In the examples, replace `myinstance.service-now.com` with the URL of your instance.

GET

```
#Need to install requests package for python
 #sudo easy_install requests
 import requests

 # Set the request parameters
 url = 'https://myinstance.service-now.com/api/now/table/incident?syspa
 user = 'admin'
 pwd = 'admin'
```

Remember session 1 - Basic GET requests with cURL ?

Raw output:

curl -v http://worldtimeapi.org/api/ip”

No parameters:

curl -v https://ipwhois.app/json/8.8.8.8 | python -m json.tool

Let’s include parameters:

curl -v https://ipwhois.app/json/8.8.8.8?objects=country,city,timezone | python -m json.tool

Single parameter:

curl https://api.exchangeratesapi.io/latest?base=AUD | python -m json.tool

Multiple parameters:

curl 'https://api.exchangeratesapi.io/latest?base=AUD&symbols=USD’ | python -m json.tool

More complex response: curl "https://endpoints.office.com/endpoints/worldwide?clientrequestid=b10c5ed1-bad1-445f-b386-b919946339a7"

Body in the request: curl -H 'Content-Type: application/json' -d '{"text": "Test message via Webhook"}' https://outlook.office.com/webhook/82e433c98-a978-465f-8254-9d541ee73c/IncomingWebhook/cb6d6416f49aaba3bd5

Making API calls using requests library

- import requests
- Define string variable - *url*
- Make request requests.<method>

like *requests.get* or *requests.post*

- Remember what we receive from

the server?

  - Response code
  - Response body
  - Headers, cookies, etc…

```python
import requests

url = 'https://api.exchangeratesapi.io/latest'

result = requests.get(url)

print(f'Got response code: {result.status_code} {result.reason} and text \n {result.text}')
```

```
requests01
C:\dev\new_project_01\venv\Scripts\python.exe C:/dev/new_project_01/requests01.py
Got response code: 200 OK  and text
 {"rates":{"CAD":1.5497,"HKD":9.2404,"ISK":159.0,"PHP":57.375,"DKK":7.4415,"HUF":362.32,"CZK":
```

- To access them, we define variable *result* – what we get from the server we put there
- Result is an object – we'll discuss objects when we learn classes
- To access various response object properties, such as content, use *<variable_name>* ***-dot-*** *<attibute>*
- The list of attributes is in the doc, IDE can also help:

result.text – response in text format

result.json () – response in json format

result.status_code, result.reason

```
print(result.)
status_code   Response
reason        Response
text          Response
url           Response
__doc__       Response
content       Response
```

Making API calls using requests library

- Adding parameters to the request: https://api.exchangeratesapi.io/latest?base=AUD&symbols=USD

^^^^^^^^^^^^ parameters

- Define dictionary

key-value pair

where

key is Parameter name

value is Parameter value

- Use argument *params* = <*your dictionary*>

```python
import requests

url = 'https://api.exchangeratesapi.io/latest'
parameters = {'base': 'AUD', 'symbols': 'USD'}

result = requests.get(url, params=parameters)

print(f'Got response code: {result.status_code} {result.reason} and text \n {result.text}')
```

requests01

```
C:\dev\new_project_01\venv\Scripts\python.exe C:/dev/new_project_01/requests01.py
Got response code: 200 OK and text
 {"rates":{"USD":0.736744531},"base":"AUD","date":"2020-11-27"}
```

```python
import requests

url = 'https://ipwhois.app/json/8.8.8.8'
parameters = {'objects': 'country,city,isp,asn'}

result = requests.get(url, params=parameters)

print(f'Got response code: {result.status_code} {result.reason} and text \n {result.text}')
```

'objects'

requests01

```
C:\dev\new_project_01\venv\Scripts\python.exe C:/dev/new_project_01/requests01.py
Got response code: 200 OK and text
 {"country":"United States","city":"Ashburn","asn":"AS15169","isp":"Google LLC"}
```

Making API calls using requests library - POST

- The example is copy-paste from ServiceNow
- The only difference is getpass – line 2 and 8

https://stackoverflow.com/questions/28579468/how-to-use-the-python-getpass-getpass-in-pycharm - getpass requires changing settings in Pycharm

- Example of importing a library which you don't have to install – it's already included with Python
- Note how we can define headers and payload (body) – dictionaries
- Use **requests.post** and specify options:

headers=, **auth=** method (basic – username and password) and **data=**

- Note we get a whole bunch of text – how to get useful data from it, like Incident number?
- Parse JSON

```python
1  import requests
2  import getpass
3
4  # Set the request parameters
5  url = 'https://dev78092.service-now.com/api/now/table/incident'
6  user = 'api_user'
7
8  pwd = getpass.getpass(prompt='Enter password: ')
9
10 # Set proper headers
11 headers = {"Content-Type": "application/json", "Accept": "application/json"}
12 # Do the HTTP request
13 response = requests.post(url, auth=(user, pwd), headers=headers, data='{"short_description":"Test from Python"}')
14
15 # Decode the JSON response into a dictionary and use the data
16 print('Status:', response.status_code, 'Headers:', response.headers, 'Response:', response.json())
```

snow_req1

```
C:\dev\new_project_01\venv\Scripts\python.exe C:/dev/new_project_01/snow_req1.py
Enter password:
Status: 201 Headers: {'Set-Cookie': 'JSESSIONID=5A0626549DE749CB8749C7963A44F3A4; Path=/; HttpOnly;Secure, glide_user=;
HttpOnly;Secure, glide_user_session=; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:10 GMT; Path=/; HttpOnly;Secure, glide_u
2147483647; Expires=Fri, 17-Dec-2088 10:28:36 GMT; Path=/; HttpOnly;Secure, glide_session_store=E3AEFB45DB3020104AC9B298
GMT; Path=/; HttpOnly;Secure, BIGipServerpool_dev78092=394418186.36926.0000; path=/; Httponly; Secure', 'Content-Encodin
'2baefb45db30', 'Location': 'https://dev78092.service-now.com/api/now/table/incident/abaefb41db3020104ac9b298f4961939',
no-store,must-revalidate,max-age=-1', 'Expires': '0', 'Content-Type': 'application/json;charset=UTF-8', 'Transfer-Encodi
Server': 'ServiceNow', 'Strict-Transport-Security': 'max-age=63072000; includeSubDomains'} Response: {'result': {'parent
'', 'upon_reject': 'cancel', 'sys_updated_on': '2020-11-29 07:14:29', 'child_incidents': '0', 'hold_reason': '', 'task_e
umber': 'INC0010010', 'resolved_by': '', 'sys_updated_by': 'api_user', 'opened_by': {'link': 'https://dev78092.service-n
1900', 'value': 'a60a84fadb5c20104ac9b298f4961900'}, 'user_input': '', 'sys_created_on': '2020-11-29 07:14:29', 'sys_dom
```

https://docs.servicenow.com/bundle/geneva-servicenow-platform/page/integrate/inbound_rest/reference/r_TableAPIPythonExamples.html

JSON

- JavaScript Object Notation
- Dataformat. Other dataformats are YAML, CSV, XML, etc.
- Used to represent complex data
- Used in REST API – de-facto data exchange format
- Elements – key : value
- Elements are separated with comma
- **JSON** is represented by Python **dictionary**
- Have a look at *address* element – it's a dictionary {}
- Have a look at *phoneNumbers* element – it's a List [] with elements which are dictionaries
- At the end of the day JSON is a string

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

https://en.wikipedia.org/wiki/JSON

Nested Data Structures – recap from Session3

Remember Lists and Dictionaries?

List of lists where elements are dictionaries and values are lists of dictionaries

data[0][0]['ip_addr'][1]['ipv4']

How we access them:

- data[0] – First element of the list *data* – we get *list_of_interfaces*
- [0] – Get element of a list with index 0 – the result is dictionary
- ["ip_addr] – Get value from dictionary using key ip_addr – this gives you another list
- [1] – get element from a list with index 1 – this gives you a dictionary
- ['ipv4'] – get value from dictionary

```python
eth1_interface_definition = {'interface_name': 'eth1',
                             'ip_addr': [
                                 {'ipv4': '10.2.3.4', 'ipv6': 'fe80::890a'},
                                 {'ipv4': '10.2.4.4', 'ipv6': 'fe80::890b'}
                             ],
                             'netmask': 25, 'active': False}

list_of_interfaces = [eth1_interface_definition]
list_of_zones = ['inside', 'outside']

data = [list_of_interfaces, list_of_zones]

print(data[0][0]['ip_addr'][1]['ipv4'])
```

'ip_addr' › 'ipv6'

file_1

```
C:\dev\test_pr1\venv\Scripts\python.exe C:/dev/test_pr1/file_1.py
10.2.4.4
```

Quiz

- What is the difference between these two lines?
- What Python data types you can see here?

```
17    eth0 = {'ipv4': '10.1.2.3', 'netmask': 24}
18    eth1 = 'ipv4:10.1.2.3, netmask:24'
```

JSON string vs Python data types

Line 17 is a Python **dictionary**

Line 18 is a **string**

```
17    eth0 = {'ipv4': '10.1.2.3', 'netmask': 24}
18    eth1 = 'ipv4:10.1.2.3, netmask:24'
```

There is a big difference between them – we can easily access the element of a **dictionary by using the key name**, for example:

eth0['ipv4'] - will get us result 10.1.2.3

To get the same result from a **string eth1**, we have to do complex string operations – split the string into multiple pieces and the check each piece until we found the matches what we need.

JSON library

- The most popular Python library to work with JSON
- Standard Python library, so you don't have to install it
- Converts Python data structures into JSON strings and back

JSON string -> Python dict, list **json.loads**

Python dict, list -> JSON string **json.dumps**

- This conversion allows to access or manipulate JSON data

Steps:

- Convert JSON string into Python data structures
- Access using standard methods for lists and dictionaries

Example of conversion JSON string to dictionary and getting element Token ---------->

https://realpython.com/python-json/

JSON library parses the strings, like this

```
198
199             if nextchar == '}':
200                 break
201             elif nextchar != ',':
```

https://github.com/python/cpython/blob/master/Lib/json/decoder.py

```
6   import json
7   cisco_dnac_sandbox_token_url = 'https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token'
8
9   token_response = requests.post(cisco_dnac_sandbox_token_url,
10                                 auth=('devnetuser','Cisco123!'),
11                                 headers={'content-type': 'application/json'},
12                                 verify=False,
13                                 )
14  text = token_response.text
15  print('This is text:              ', text)
16
17  response_as_dict = json.loads(token_response.text)
18  print('This is dictionary element:', response_as_dict['Token'])
19
```

requests_dna_center01

```
C:\dev\new_project_01\venv\Scripts\python.exe C:/dev/new_project_01/requests_dna_center01.py
This is text:              {"Token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI1ZTlkYmI3
This is dictionary element: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI1ZTlkYmI3NzdjZDQ3ZT
```

Parsing JSON data

- We send a GET request and received some data – it's JSON string, we convert it into Python data types
- In the response we get *{"response":[{"memorySize":"NA",serialNumber":"FCW2136L0AK",....etc* - how to get Serial number?

1 .Notice *{"response":* - it means it is a dictionary, so we can use **key** *response*

2. Notice {"response":[- it means the **value** is a list, so we need to use **indexes**, like 0

3. Notice {"response":[{- it means it is a dictionary again, and we use key value *serialNumber*

Final result - json_data['response'][0]['serialNumber']

```
response_as_dict = json.loads(token_response.text)
token = response_as_dict['Token']
# token = json.loads(token_response.text)['Token']

response = requests.get('https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device',
                        headers={'X-Auth-Token': token, 'Content-type': 'application/json'})
json_data = json.loads(response.text)

print('Raw string:                    ', response.text)
print('Whole response as dict:        ', json_data)
print('Response element:              ', json_data['response'])
print('First element of list:         ', json_data['response'][0])
print('Get SN using serialNumber key:', json_data['response'][0]['serialNumber'])
```

requests_dna_center01

```
C:\dev\api_python_training\venv\Scripts\python.exe C:/dev/api_python_training/requests_dna_center01.py
Raw string:                    {"response":[{"memorySize":"NA","family":"Switches and Hubs","hostname":"cat_9k_1","macAddress":"f8:7b:20:67:62:80","serialNumber":"FCW2136L0AK",
Whole response as dict:        {'response': [{'memorySize': 'NA', 'family': 'Switches and Hubs', 'hostname': 'cat_9k_1', 'macAddress': 'f8:7b:20:67:62:80', 'serialNumber': 'FCW
Response element:              [{'memorySize': 'NA', 'family': 'Switches and Hubs', 'hostname': 'cat_9k_1', 'macAddress': 'f8:7b:20:67:62:80', 'serialNumber': 'FCW2136L0AK', 'i
First element of list:         {'memorySize': 'NA', 'family': 'Switches and Hubs', 'hostname': 'cat_9k_1', 'macAddress': 'f8:7b:20:67:62:80', 'serialNumber': 'FCW2136L0AK', 'in
Get SN using serialNumber key: FCW2136L0AK
```

Converting JSON data back to string

- In API requests you send JSON as a string, not Python dicts/lists, so need to convert complex data types into string
- This procedure is called serialization
- Usage - *json.dumps(some_dictionary)*
- Very useful for pretty printing responses

- Note there is command *json.dump*

and json.load **without s**

used to read/write JSON into a file

- As usual, check the documentation for full

list of commands

https://www.geeksforgeeks.org/python-difference-between-json-dump-and-json-dumps/

- Pretty printing JSON response

```python
response = requests.get('https://sandboxdnac.cisco.com/dna/intent/api/v1/network-device',
                        headers={'X-Auth-Token': token, 'Content-type': 'application/json'})
# json_data is a dictionary
json_data = json.loads(response.text)
# Convert to a string, and pretty print
print(json.dumps(json_data, sort_keys=True, indent=4))
# OR in a single string:
print(json.dumps(json.loads(response.text), sort_keys=True, indent=4))
```

requests_dna_center01


```
{
    "response": [
        {
            "apManagerInterfaceIp": "",
            "associatedWlcIp": "",
            "bootDateTime": "2020-08-17 07:31:19",
            "collectionInterval": "Global Default",
            "collectionStatus": "Managed",
            "deviceSupportLevel": "Supported",
            "errorCode": null,
            "errorDescription": null,
            "family": "Switches and Hubs",
            "hostname": "cat_9k_1",
```

Demo

Summary and next steps

- Summary

Python – installing libraries, request library, JSON library, made API calls

- Homework

Try different API calls, access JSON data in responses

https://docs.servicenow.com/bundle/geneva-servicenow-platform/page/integrate/inbound_rest/reference/r_TableAPIPythonExamples.html

Try different requests from session 1 and 2 – add parameters, headers, authorization information

- Next time

Python control structures – if, for, while

Handling exceptions

Functions