**Content**

[1. JSON and data types 2](#_Toc15989685)

[1. Intro into JSON 2](#_Toc15989686)

[2. Convert from JSON to Python 2](#_Toc15989687)

[3. Convert from Python to Json 3](#_Toc15989688)

[2. Iterators, yield, generators 4](#_Toc15989689)

[1. Generator functions 4](#_Toc15989690)

[2. Generator expressions 5](#_Toc15989691)

[3. Libs 6](#_Toc15989692)

[1. Troposphere - AWS 6](#_Toc15989693)

[2. Scrapy 6](#_Toc15989694)

[3. Pendulum (something like Arrow) 6](#_Toc15989695)

[4. Requests 6](#_Toc15989696)

[5. PyFlux 6](#_Toc15989697)

[6. Zappa – AWS 7](#_Toc15989698)

[7. Others 7](#_Toc15989699)

1. What is it?

* Nbconvert to convert .ipnb?
* HTTPBasicAuth
* Other authentication methods in python ?!

1. JSON and data types
   1. Intro into JSON

In computing, **JavaScript Object Notation (JSON)** is an **open-standard file format** that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format used for asynchronous browser–server communication, including as a replacement for XML in some AJAX-style systems.

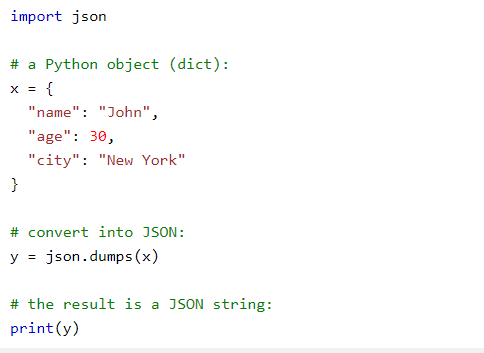
JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. The official Internet media type for JSON is application/json.

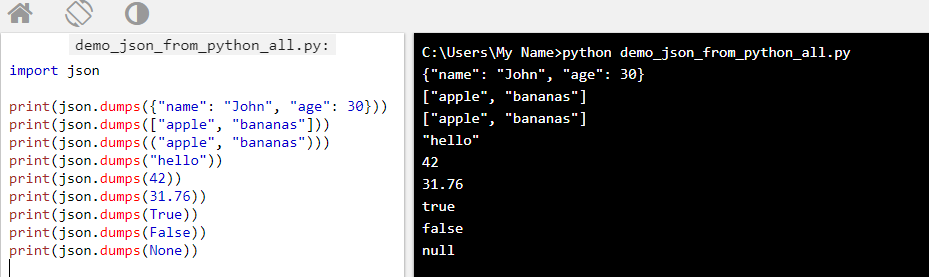
**Asynchronous JavaScript and JSON (or AJAJ)** refers to the same dynamic web page methodology as **Ajax**, but instead of **XML**, **JSON** is the data format. AJAJ is a web development technique that provides for the ability of a webpage to request new data after it has loaded into the web browser. Typically it renders new data from the server in response to user actions on that webpage. For example, what the user types into a search box, client-side code then sends to the server, which immediately responds with a drop-down list of matching database items.

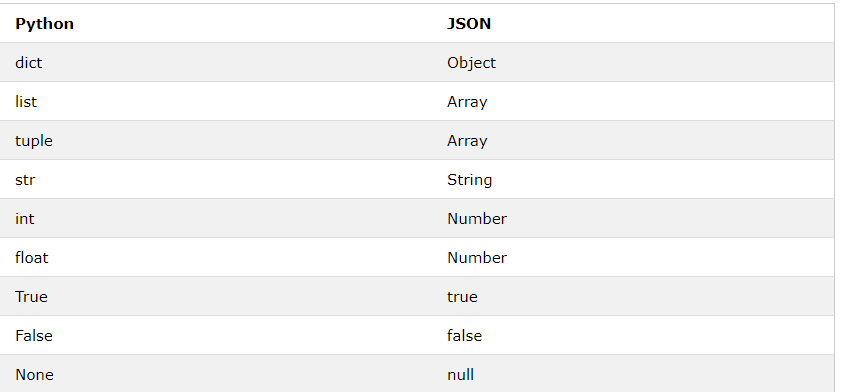
* 1. Convert from JSON to Python



* 1. Convert from Python to Json





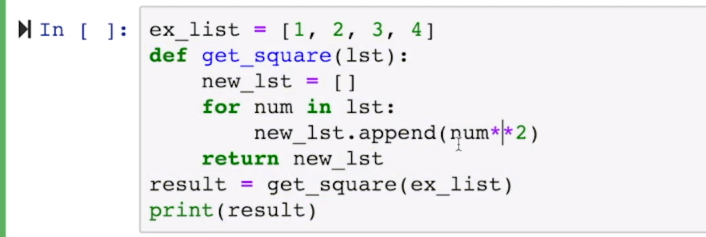


1. Iterators, yield, generators

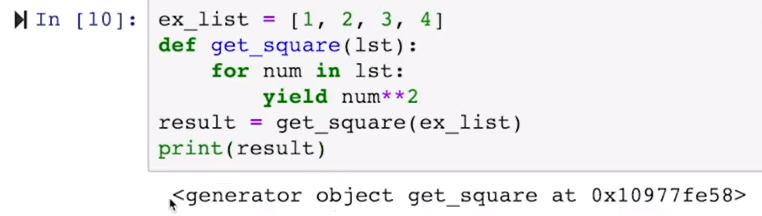
* Generator functions (yield)
* Generator expressions
* Optimizing the memory space.
* E.g. when working w large excel to not run out of memory (Lol 2 late  )
* Result set need not be constructed all at once
  1. Generator functions

Unlike normal functions, that receive an input and return a result immediately, generator functions produce result only when needed.

Python generators are able to suspend and resume their execution around the point of value generation



But we do not need the temporary list.



3 ways how to go through a generator object:

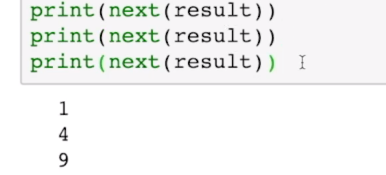
1. For loop



1. list built-in method to convert the iterator to a list

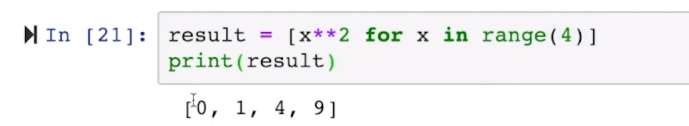


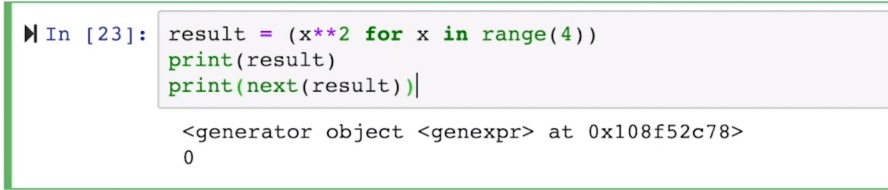
1. next method: every iterator object in python defines a next method. Returns the next item in the iteration till all of the items are over. Results in StopIteration error.



* 1. Generator expressions

List comprehension:





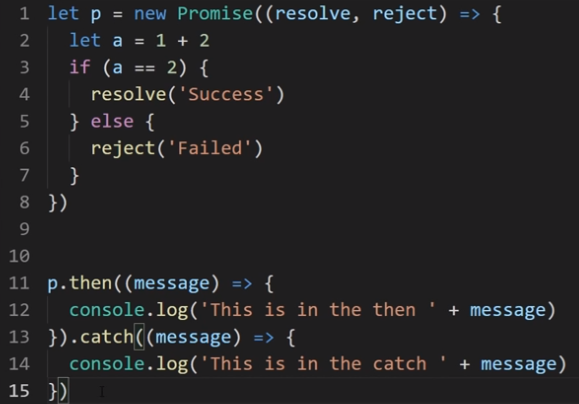
1. Async/await and friends

**Concurrent programming** in dynamic languages like Python and Javascript has become a requirement if you want to write any sort of decently performing code doing any kind of **significant** **IO** (disk or network).

Dynamic languages like Python and Javascript pretty much suck at executing code in parallel (what’s called parallelism, or more precisely code parallelism). Both are practically barred from doing it via threads: Python’s GIL and Javascript’s intentionally single-threaded design make threads-based parallelism (almost) impossible. And doing it via processes makes it equivalent to running multiple programs (or copies of the same program) plus handling inter-process communication between then — this is both complex and resource (RAM) hungry.

* 1. Promise

**Promise** in JS is like promise in real life, what you do is you commit to something by saying I promise to do something, for example I promise to make the best cake ever, and then that promise either has 2 results, either that promise has **completed**, it is **resolved (then)** or it is **failed**, and it is **rejected (catch)**.



* 1. Future

1. Libs
   1. Troposphere - AWS

The main benefit of Troposphere is that it can create dynamic CloudFormation templates. For example, if we need to get some user input (Region, AMI, OS types etc.) which can be dynamic and based on that we need to create CloudFormation template, in this scenario troposphere is very handy. We can create reusable Python code using troposphere and AWS Python SDK (boto3) to generate dynamic CloudFormation template based on user input. Other than this, troposphere has some other benefits over CloudFormation.

* 1. Scrapy

Scrappy is widely used Python web scraping library. It is used for creating crawling programs. Initially, it was designed for scraping, like its name indicate but now it used for many purposes including data mining, automated testing, etc. scrapy is open-source and must have library.

* 1. Pendulum (something like Arrow)

Pendulum is a python package which is used to determine pendulum. It make life a lot of easier when it comes to work with date and time. You code will still work if you replace every elements of DateTime. With Pendulum, you can parse DateTime, and display datetime with time zone. So basically Pendulum is improved version of Arrow library and it have all the handy methods like rounding, truncating, converting, parsing, formatting, and arithmetic.

* 1. Requests

Requests is one of the famous Python library which is licensed under Apache2 and written in Python. This library help humans to interact with the languages. With Request library, you don’t need to add query, string manually to your URL’s or form-encode your POST data. You can send HTTP request to server using Request library and you can add form data, content like header, multi-part files, etc.

* 1. PyFlux

Pyflux is a python library which is used to predict and analysis time series. It is developed by Ross Taylor, this library have many options for interface and contain many new classes of model types. Pyflux allow users to implement many modern time series models like GARCH and predict the nature that how it will react in future.

* 1. Zappa – AWS

Zappa is one of best python package which is created by Miserlou, it so easy to build and implement server-less application on API Gateway and Amazon Web Services Lambda. Since AWS handling the horizontal scaling automatically, so no request going to be time out. With Zappa, you can update your code in single line with Zappa.

e.g. django-zappa : Serverless Django w AWS Lambda + API Gateway

* 1. Others
* **Theano**
* **IPython**
* **Imbalanced-learn**
* **Caffe2**
* **Dash**
* **Fire**
* **Flashtext**
* **Pipenv** – working w dependencies
* **Luminoth** – py toolkit for computer vision
* **Tqdm** – progress bars, debugging for loops
* **Pillow** - The Python Imaging Library adds image processing capabilities to your Python interpreter. Image Processing capabilities - The library contains basic image processing functionality, including point operations, filtering with a set of built-in convolution kernels, and colour space conversions. The library also supports image resizing, rotation and arbitrary affine transforms. There’s a histogram method allowing you to pull some statistics out of an image. This can be used for automatic contrast enhancement, and for global statistical analysis.
* **Scapy** - It can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery (it can replace *hping*, 85% of *nmap, arpspoof, arp-sk, arping, tcpdump, wireshark, p0f,* etc.). It also performs very well at a lot of other specific tasks that most other tools can't handle, like sending invalid frames, injecting your own 802.11 frames, combining techniques (VLAN hopping+ARP cache poisoning, VoIP decoding on WEP protected channel, ...), etc.
* **Nltk** – Natural Language Toolkit
* **SQLAlchemy** – Database toolkit, persistence
* **Twisted** – networking framework
* **Click** - Click is a Python package for creating beautiful command line interfaces in a composable way with as little code as necessary. It’s the “Command Line Interface Creation Kit”. It’s highly configurable but comes with sensible defaults out of the box.

1. Python Environments