Hands-on Lab: API Examples

Random User and Fruityvice API Examples

Estimated time needed: **30** minutes

Objectives

After completing this lab you will be able to:

- Load and use RandomUser API, using `RandomUser()` Python library
- Load and use Fruityvice API, using `requests` Python library
- Load and use Open-Joke-API, using `requests` Python library

The purpose of this notebook is to provide more examples on how to use simple APIs. As you have already learned from previous videos and notebooks, API stands for Application Programming Interface and is a software intermediary that allows two applications to talk to each other.

The advantages of using APIs:

- **Automation**. Less human effort is required and workflows can be easily updated to become faster and more
  productive.
- **Efficiency**. It allows to use the capabilities of one of the already developed APIs than to try to independently implement some functionality from scratch.

The disadvantage of using APIs:

- **Security**. If the API is poorly integrated, it means it will be vulnerable to attacks, resulting in data breeches or losses having financial or reputation implications.

One of the applications we will use in this notebook is Random User Generator. RandomUser is an open-source, free API providing developers with randomly generated users to be used as placeholders for testing purposes. This makes the tool similar to Lorem Ipsum, but is a placeholder for people instead of text. The API can return multiple results, as well as specify generated user details such as gender, email, image, username, address, title, first and last name, and more. More information on RandomUser can be found here.

Another example of simple API we will use in this notebook is Fruityvice application. The Fruityvice API web service which provides data for all kinds of fruit! You can use Fruityvice to find out interesting information about fruit and educate yourself. The web service is completely free to use and contribute to.

Example 1: RandomUser API

Bellow are Get Methods parameters that we can generate. For more information on the parameters, please visit this documentation page.

Get Methods

- get_cell()
- get_city()
- get_dob()
- get_email()
- get_first_name()
- get_full_name()
- get_gender()
- get_id()
- get_id_number()
- get_id_type()
- get_info()
- get_last_name()
- get_login_md5()
- get_login_salt()
- get_login_sha1()
- get_login_sha256()
- get_nat()
- get_password()
- get_phone()
- get_picture()
- get_postcode()
- get_registered()
- get_state()
- get_street()
- get_username()
- get_zipcode()

To start using the API you can install the `randomuser` library running the `pip install` command.

```
!pip install randomuser

Collecting randomuser
  Downloading randomuser-1.6.tar.gz (5.0 kB)
  Preparing metadata (setup.py) ... user
  Building wheel for randomuser (setup.py) ... user:
filename=randomuser-1.6-py3-none-any.whl size=5066
sha256=d8a782cd7dbfdd78a09f48698f77742716b298cda8241117ecf8282d6d05f25
6
  Stored in directory:
/home/jupyterlab/.cache/pip/wheels/21/10/7b/c13bc3e24a3c1000a34554802a
d8aa0ff27df56366998a0473
Successfully built randomuser
```

```
Installing collected packages: randomuser
Successfully installed randomuser-1.6
```

Then, we will load the necessary libraries.

```
from randomuser import RandomUser
import pandas as pd
```

First, we will create a random user object, r.

```
r = RandomUser()
```

Then, using `generate_users()` function, we get a list of random 10 users.

```
some_list = r.generate_users(10)

some_list
```

```
[<randomuser.RandomUser at 0x7f83dfcae050>,
 <randomuser.RandomUser at 0x7f83dfcae090>,
 <randomuser.RandomUser at 0x7f83dfcae0d0>,
 <randomuser.RandomUser at 0x7f83dfcae110>,
 <randomuser.RandomUser at 0x7f83dfcae150>,
 <randomuser.RandomUser at 0x7f83dfcae1d0>,
 <randomuser.RandomUser at 0x7f83dfcae210>,
 <randomuser.RandomUser at 0x7f83dfcae250>,
 <randomuser.RandomUser at 0x7f83dfcae290>,
 <randomuser.RandomUser at 0x7f83dfcae190>]
```

The **"Get Methods"** functions mentioned at the beginning of this notebook, can generate the required parameters to construct a dataset. For example, to get full name, we call `get_full_name()` function.

```
name = r.get_full_name()
```

Let's say we only need 10 users with full names and their email addresses. We can write a "for-loop" to print these 10 users.

```
for user in some_list:
    print (user.get_full_name()," ",user.get_email())
```

```
Eileen Ophus   eileen.ophus@example.com
Ilya Tollenaar   ilya.tollenaar@example.com
Christoph Eggers   christoph.eggers@example.com
Zoe Peterson   zoe.peterson@example.com
Joan Morales   joan.morales@example.com
Deniz Kolman   deniz.kolman@example.com
Annabell Østlie   annabell.ostlie@example.com
```

```
Thaïs Masson   thais.masson@example.com
Karla Rasmussen   karla.rasmussen@example.com
Florence Moore   florence.moore@example.com
```

Exercise 1

In this Exercise, generate photos of the random 10 users.

```
## Write your code here
for user in some_list:
    print (user.get_picture())
```

```
https://randomuser.me/api/portraits/women/18.jpg
https://randomuser.me/api/portraits/women/58.jpg
https://randomuser.me/api/portraits/men/64.jpg
https://randomuser.me/api/portraits/women/57.jpg
https://randomuser.me/api/portraits/men/84.jpg
https://randomuser.me/api/portraits/men/70.jpg
https://randomuser.me/api/portraits/women/64.jpg
https://randomuser.me/api/portraits/women/11.jpg
https://randomuser.me/api/portraits/women/45.jpg
https://randomuser.me/api/portraits/women/30.jpg
```

To generate a table with information about the users, we can write a function containing all desirable parameters. For example, name, gender, city, etc. The parameters will depend on the requirements of the test to be performed. We call the Get Methods, listed at the beginning of this notebook. Then, we return pandas dataframe with the users.

```
def get_users():
    users =[]

    for user in RandomUser.generate_users(10):

users.append({"Name":user.get_full_name(),"Gender":user.get_gender(),"
City":user.get_city(),"State":user.get_state(),"Email":user.get_email(
), "DOB":user.get_dob(),"Picture":user.get_picture()})

    return pd.DataFrame(users)

get_users()
```

```
                    Name  Gender             City
State  \
0          Baptiste Joly    male      Glarus Nord
Obwalden
1           پرهام کریمی   male         کرمانشاه         قائم‌شهر
2      Sebastian Porter    male           Galway
Wexford
3         Oliver Ambrose    male           Borden
```

```
Québec
4         Warren Clarke    male       Birmingham  Dumfries and
Galloway
5           Troy Turner    male            Tweed
Victoria
6  Kristoffer Bjørkhaug    male            Kolbu          Troms -
Romsa
7       Ingridt Peixoto  female          Itatiba
Amazonas
8            Aloïs Roux    male         Grenoble
Somme
9     Agathe Carpentier  female  Vitry-sur-Seine
Orne

                              Email                          DOB  \
0         baptiste.joly@example.com  1977-06-08T17:22:30.264Z
1           prhm.khrymy@example.com  1984-05-17T07:17:50.833Z
2      sebastian.porter@example.com  1984-06-24T21:55:05.729Z
3        oliver.ambrose@example.com  1960-04-10T04:17:59.840Z
4         warren.clarke@example.com  1963-10-01T22:51:04.229Z
5           troy.turner@example.com  1972-09-03T09:45:11.363Z
6  kristoffer.bjorkhaug@example.com  1980-10-24T17:05:40.049Z
7       ingridt.peixoto@example.com  1952-08-15T05:14:55.041Z
8            alois.roux@example.com  1980-07-06T05:12:02.338Z
9     agathe.carpentier@example.com  1974-05-25T12:41:22.817Z

                                          Picture
0    https://randomuser.me/api/portraits/men/45.jpg
1    https://randomuser.me/api/portraits/men/77.jpg
2    https://randomuser.me/api/portraits/men/89.jpg
3    https://randomuser.me/api/portraits/men/82.jpg
4    https://randomuser.me/api/portraits/men/49.jpg
5    https://randomuser.me/api/portraits/men/95.jpg
6    https://randomuser.me/api/portraits/men/63.jpg
7  https://randomuser.me/api/portraits/women/58.jpg
8    https://randomuser.me/api/portraits/men/75.jpg
9  https://randomuser.me/api/portraits/women/27.jpg

df1 = pd.DataFrame(get_users())
```

Now we have a *pandas* dataframe that can be used for any testing purposes that the tester might have.

Example 2: Fruityvice API

Another, more common way to use APIs, is through `requests` library. The next lab, Requests and HTTP, will contain more information about requests.

We will start by importing all required libraries.

```
import requests
import json
```

We will obtain the fruityvice API data using `requests.get("url")` function. The data is in a json format.

```
data = requests.get("https://fruityvice.com/api/fruit/all")
```

We will retrieve results using `json.loads()` function.

```
results = json.loads(data.text)
```

We will convert our json data into *pandas* data frame.

```
pd.DataFrame(results)
```

```
                  name   id           family              order
genus  \
0            Persimmon   52        Ebenaceae            Rosales
Diospyros
1           Strawberry    3         Rosaceae            Rosales
Fragaria
2               Banana    1         Musaceae      Zingiberales
Musa
3               Tomato    5       Solanaceae         Solanales
Solanum
4                 Pear    4         Rosaceae            Rosales
Pyrus
5               Durian   60        Malvaceae          Malvales
Durio
6           Blackberry   64         Rosaceae            Rosales
Rubus
7          Lingonberry   65        Ericaceae          Ericales
Vaccinium
8                 Kiwi   66    Actinidiaceae  Struthioniformes
Apteryx
9               Lychee   67      Sapindaceae        Sapindales
Litchi
10           Pineapple   10     Bromeliaceae            Poales
Ananas
11                 Fig   68         Moraceae            Rosales
Ficus
12          Gooseberry   69  Grossulariaceae      Saxifragales
Ribes
13        Passionfruit   70   Passifloraceae      Malpighiales
Passiflora
14                Plum   71         Rosaceae            Rosales
Prunus
```

15	Orange	2	Rutaceae	Sapindales	Citrus
16	GreenApple	72	Rosaceae	Rosales	Malus
17	Raspberry	23	Rosaceae	Rosales	Rubus
18	Watermelon	25	Cucurbitaceae	Cucurbitales	Citrullus
19	Lemon	26	Rutaceae	Sapindales	Citrus
20	Mango	27	Anacardiaceae	Sapindales	Mangifera
21	Blueberry	33	Rosaceae	Rosales	Fragaria
22	Apple	6	Rosaceae	Rosales	Malus
23	Guava	37	Myrtaceae	Myrtales	Psidium
24	Apricot	35	Rosaceae	Rosales	Prunus
25	Melon	41	Cucurbitaceae	Cucurbitaceae	Cucumis
26	Tangerine	77	Rutaceae	Sapindales	Citrus
27	Pitahaya	78	Cactaceae	Caryophyllales	Cactaceae
28	Lime	44	Rutaceae	Sapindales	Citrus
29	Pomegranate	79	Lythraceae	Myrtales	Punica
30	Dragonfruit	80	Cactaceae	Caryophyllales	Selenicereus
31	Grape	81	Vitaceae	Vitales	Vitis
32	Morus	82	Moraceae	Rosales	Morus
33	Feijoa	76	Myrtaceae	Myrtoideae	Sellowiana
34	Avocado	84	Lauraceae	Laurales	Persea
35	Kiwifruit	85	Actinidiaceae	Ericales	Actinidia
36	Cranberry	87	Ericaceae	Ericales	Vaccinium
37	Cherry	9	Rosaceae	Rosales	Prunus
38	Peach	86	Rosaceae	Rosales	Prunus
39	Jackfruit	94	Moraceae	Rosales	

```
Artocarpus
40        Horned Melon   95    Cucurbitaceae      Cucurbitales
Cucumis
41            Hazelnut   96       Betulaceae           Fagales
Corylus
42              Pomelo   98         Rutaceae        Sapindales
Citrus
43          Mangosteen   99       Clusiaceae      Malpighiales
Garcinia
44             Pumpkin  100    Cucurbitaceae      Cucurbitales
Cucurbita
45  Japanese Persimmon  101        Ebenaceae          Ericales
Diospyros
46              Papaya   42       Caricaceae       Brassicales
Carica
47              Annona  103       Annonaceae           Rosales
Annonas
48   Ceylon Gooseberry  104       Salicaceae      Malpighiales
Dovyalis

                                             nutritions
0   {'calories': 81, 'fat': 0.0, 'sugar': 18.0, 'c...
1   {'calories': 29, 'fat': 0.4, 'sugar': 5.4, 'ca...
2   {'calories': 96, 'fat': 0.2, 'sugar': 17.2, 'c...
3   {'calories': 74, 'fat': 0.2, 'sugar': 2.6, 'ca...
4   {'calories': 57, 'fat': 0.1, 'sugar': 10.0, 'c...
5   {'calories': 147, 'fat': 5.3, 'sugar': 6.75, '...
6   {'calories': 40, 'fat': 0.4, 'sugar': 4.5, 'ca...
7   {'calories': 50, 'fat': 0.34, 'sugar': 5.74, '...
8   {'calories': 61, 'fat': 0.5, 'sugar': 9.0, 'ca...
9   {'calories': 66, 'fat': 0.44, 'sugar': 15.0, '...
10  {'calories': 50, 'fat': 0.12, 'sugar': 9.85, '...
11  {'calories': 74, 'fat': 0.3, 'sugar': 16.0, 'c...
12  {'calories': 44, 'fat': 0.6, 'sugar': 0.0, 'ca...
13  {'calories': 97, 'fat': 0.7, 'sugar': 11.2, 'c...
14  {'calories': 46, 'fat': 0.28, 'sugar': 9.92, '...
15  {'calories': 43, 'fat': 0.2, 'sugar': 8.2, 'ca...
16  {'calories': 21, 'fat': 0.1, 'sugar': 6.4, 'ca...
17  {'calories': 53, 'fat': 0.7, 'sugar': 4.4, 'ca...
18  {'calories': 30, 'fat': 0.2, 'sugar': 6.0, 'ca...
19  {'calories': 29, 'fat': 0.3, 'sugar': 2.5, 'ca...
20  {'calories': 60, 'fat': 0.38, 'sugar': 13.7, '...
21  {'calories': 29, 'fat': 0.4, 'sugar': 5.4, 'ca...
22  {'calories': 52, 'fat': 0.4, 'sugar': 10.3, 'c...
23  {'calories': 68, 'fat': 1.0, 'sugar': 9.0, 'ca...
24  {'calories': 15, 'fat': 0.1, 'sugar': 3.2, 'ca...
25  {'calories': 34, 'fat': 0.0, 'sugar': 8.0, 'ca...
26  {'calories': 45, 'fat': 0.4, 'sugar': 9.1, 'ca...
27  {'calories': 36, 'fat': 0.4, 'sugar': 3.0, 'ca...
```

```
28  {'calories': 25, 'fat': 0.1, 'sugar': 1.7, 'ca...
29  {'calories': 83, 'fat': 1.2, 'sugar': 13.7, 'c...
30  {'calories': 60, 'fat': 1.5, 'sugar': 8.0, 'ca...
31  {'calories': 69, 'fat': 0.16, 'sugar': 16.0, '...
32  {'calories': 43, 'fat': 0.39, 'sugar': 8.1, 'c...
33  {'calories': 44, 'fat': 0.4, 'sugar': 3.0, 'ca...
34  {'calories': 160, 'fat': 14.66, 'sugar': 0.66,...
35  {'calories': 61, 'fat': 0.5, 'sugar': 8.9, 'ca...
36  {'calories': 46, 'fat': 0.1, 'sugar': 4.0, 'ca...
37  {'calories': 50, 'fat': 0.3, 'sugar': 8.0, 'ca...
38  {'calories': 39, 'fat': 0.25, 'sugar': 8.4, 'c...
39  {'calories': 95, 'fat': 0.0, 'sugar': 19.1, 'c...
40  {'calories': 44, 'fat': 1.26, 'sugar': 0.5, 'c...
41  {'calories': 628, 'fat': 61.0, 'sugar': 4.3, '...
42  {'calories': 37, 'fat': 0.0, 'sugar': 8.5, 'ca...
43  {'calories': 73, 'fat': 0.58, 'sugar': 16.11, ...
44  {'calories': 25, 'fat': 0.3, 'sugar': 3.3, 'ca...
45  {'calories': 70, 'fat': 0.2, 'sugar': 13.0, 'c...
46  {'calories': 39, 'fat': 0.3, 'sugar': 4.4, 'ca...
47  {'calories': 92, 'fat': 0.29, 'sugar': 3.4, 'c...
48  {'calories': 47, 'fat': 0.3, 'sugar': 8.1, 'ca...
```

The result is in a nested json format. The 'nutrition' column contains multiple subcolumns, so the data needs to be 'flattened' or normalized.

```
df2 = pd.json_normalize(results)

df2
```

```
                   name   id           family              order
genus  \
0            Persimmon   52        Ebenaceae           Rosales
Diospyros
1           Strawberry    3         Rosaceae           Rosales
Fragaria
2               Banana    1         Musaceae      Zingiberales
Musa
3               Tomato    5       Solanaceae         Solanales
Solanum
4                 Pear    4         Rosaceae           Rosales
Pyrus
5               Durian   60        Malvaceae          Malvales
Durio
6           Blackberry   64         Rosaceae           Rosales
Rubus
7          Lingonberry   65        Ericaceae          Ericales
Vaccinium
8                 Kiwi   66    Actinidiaceae  Struthioniformes
Apteryx
```

9	Lychee	67	Sapindaceae	Sapindales	Litchi
10	Pineapple	10	Bromeliaceae	Poales	Ananas
11	Fig	68	Moraceae	Rosales	Ficus
12	Gooseberry	69	Grossulariaceae	Saxifragales	Ribes
13	Passionfruit	70	Passifloraceae	Malpighiales	Passiflora
14	Plum	71	Rosaceae	Rosales	Prunus
15	Orange	2	Rutaceae	Sapindales	Citrus
16	GreenApple	72	Rosaceae	Rosales	Malus
17	Raspberry	23	Rosaceae	Rosales	Rubus
18	Watermelon	25	Cucurbitaceae	Cucurbitales	Citrullus
19	Lemon	26	Rutaceae	Sapindales	Citrus
20	Mango	27	Anacardiaceae	Sapindales	Mangifera
21	Blueberry	33	Rosaceae	Rosales	Fragaria
22	Apple	6	Rosaceae	Rosales	Malus
23	Guava	37	Myrtaceae	Myrtales	Psidium
24	Apricot	35	Rosaceae	Rosales	Prunus
25	Melon	41	Cucurbitaceae	Cucurbitaceae	Cucumis
26	Tangerine	77	Rutaceae	Sapindales	Citrus
27	Pitahaya	78	Cactaceae	Caryophyllales	Cactaceae
28	Lime	44	Rutaceae	Sapindales	Citrus
29	Pomegranate	79	Lythraceae	Myrtales	Punica
30	Dragonfruit	80	Cactaceae	Caryophyllales	Selenicereus
31	Grape	81	Vitaceae	Vitales	Vitis
32	Morus	82	Moraceae	Rosales	Morus
33	Feijoa	76	Myrtaceae	Myrtoideae	

```
Sellowiana
34             Avocado   84        Lauraceae          Laurales
Persea
35           Kiwifruit   85    Actinidiaceae          Ericales
Actinidia
36           Cranberry   87        Ericaceae          Ericales
Vaccinium
37              Cherry    9         Rosaceae           Rosales
Prunus
38               Peach   86         Rosaceae           Rosales
Prunus
39           Jackfruit   94         Moraceae           Rosales
Artocarpus
40        Horned Melon   95    Cucurbitaceae      Cucurbitales
Cucumis
41            Hazelnut   96       Betulaceae           Fagales
Corylus
42              Pomelo   98         Rutaceae        Sapindales
Citrus
43          Mangosteen   99       Clusiaceae      Malpighiales
Garcinia
44             Pumpkin  100    Cucurbitaceae      Cucurbitales
Cucurbita
45  Japanese Persimmon  101        Ebenaceae          Ericales
Diospyros
46              Papaya   42       Caricaceae       Brassicales
Carica
47              Annona  103       Annonaceae           Rosales
Annonas
48   Ceylon Gooseberry  104       Salicaceae      Malpighiales
Dovyalis

    nutritions.calories  nutritions.fat  nutritions.sugar  \
0                    81            0.00             18.00
1                    29            0.40              5.40
2                    96            0.20             17.20
3                    74            0.20              2.60
4                    57            0.10             10.00
5                   147            5.30              6.75
6                    40            0.40              4.50
7                    50            0.34              5.74
8                    61            0.50              9.00
9                    66            0.44             15.00
10                   50            0.12              9.85
11                   74            0.30             16.00
12                   44            0.60              0.00
13                   97            0.70             11.20
14                   46            0.28              9.92
15                   43            0.20              8.20
```

16	21	0.10	6.40
17	53	0.70	4.40
18	30	0.20	6.00
19	29	0.30	2.50
20	60	0.38	13.70
21	29	0.40	5.40
22	52	0.40	10.30
23	68	1.00	9.00
24	15	0.10	3.20
25	34	0.00	8.00
26	45	0.40	9.10
27	36	0.40	3.00
28	25	0.10	1.70
29	83	1.20	13.70
30	60	1.50	8.00
31	69	0.16	16.00
32	43	0.39	8.10
33	44	0.40	3.00
34	160	14.66	0.66
35	61	0.50	8.90
36	46	0.10	4.00
37	50	0.30	8.00
38	39	0.25	8.40
39	95	0.00	19.10
40	44	1.26	0.50
41	628	61.00	4.30
42	37	0.00	8.50
43	73	0.58	16.11
44	25	0.30	3.30
45	70	0.20	13.00
46	39	0.30	4.40
47	92	0.29	3.40
48	47	0.30	8.10

	nutritions.carbohydrates	nutritions.protein
0	18.00	0.00
1	5.50	0.80
2	22.00	1.00
3	3.90	0.90
4	15.00	0.40
5	27.10	1.50
6	9.00	1.30
7	11.30	0.75
8	15.00	1.10
9	17.00	0.80
10	13.12	0.54
11	19.00	0.80
12	10.00	0.90
13	22.40	2.20

```
14                      11.40                    0.70
15                       8.30                    1.00
16                       3.10                    0.40
17                      12.00                    1.20
18                       8.00                    0.60
19                       9.00                    1.10
20                      15.00                    0.82
21                       5.50                    0.00
22                      11.40                    0.30
23                      14.00                    2.60
24                       3.90                    0.50
25                       8.00                    0.00
26                       8.30                    0.00
27                       7.00                    1.00
28                       8.40                    0.30
29                      18.70                    1.70
30                       9.00                    9.00
31                      18.10                    0.72
32                       9.80                    1.44
33                       8.00                    0.60
34                       8.53                    2.00
35                      14.60                    1.14
36                      12.20                    0.40
37                      12.00                    1.00
38                       9.50                    0.90
39                      23.20                    1.72
40                       7.56                    1.78
41                      17.00                   15.00
42                       9.67                    0.82
43                      17.91                    0.41
44                       4.60                    1.10
45                      19.00                    0.60
46                       5.80                    0.50
47                      19.10                    1.50
48                       9.60                    1.20
```

Let's see if we can extract some information from this dataframe. Perhaps, we need to know the family and genus of a cherry.

```
cherry = df2.loc[df2["name"] == 'Cherry']
(cherry.iloc[0]['family']) , (cherry.iloc[0]['genus'])
```

```
('Rosaceae', 'Prunus')
```

Exercise 2

In this Exercise, find out how many calories are contained in a banana.

```
# Write your code here
cal_banana = df2.loc[df2["name"] == 'Banana']
cal_banana.iloc[0]['nutritions.calories']
```

```
96
```

Exercise 3

This page contains a list of free public APIs for you to practice. Let us deal with the following example.

Official Joke API

This API returns random jokes from a database. The following URL can be used to retrieve 10 random jokes.

https://official-joke-api.appspot.com/jokes/ten

1. Using `requests.get("url")` function, load the data from the URL.

```
# Write your code here
data2 =
requests.get("https://official-joke-api.appspot.com/jokes/ten")
```

1. Retrieve results using `json.loads()` function.

```
# Write your code here
results2 = json.loads(data2.text)
```

1. Convert json data into *pandas* data frame. Drop the type and id columns.

```
# Write your code here
df3 = pd.DataFrame(results2)
df3.drop(columns=["type","id"],inplace=True)
df3
```

```
                                                 setup  \
0  Did you hear about the runner who was criticized?
1  Why did the invisible man turn down the job of...
2                  How does a dyslexic poet write?
3  What's the difference between a hippo and a zi...
4  What happens to a frog's car when it breaks down?
5          What's orange and sounds like a parrot?
6  What do you call a fashionable lawn statue wit...
7             What kind of dinosaur loves to sleep?
8         Why does Superman get invited to dinners?
9             What do you call a cow with two legs?

                                       punchline
0                        He just took it in stride
1                He couldn't see himself doing it
```

```
2                                                Inverse.
3  One is really heavy, the other is a little lig...
4                                    It gets toad away
5                                            A Carrot.
6                                        A metro-gnome
7                                    A stega-snore-us.
8                          Because he is a Supperhero.
9                                         Lean beef.
```

Congratulations! - You have completed the lab

Author

Svitlana Kramar

Svitlana is a master's degree Data Science and Analytics student at University of Calgary, who enjoys travelling, learning new languages and cultures and loves spreading her passion for Data Science.

Additional Contributor

Abhishek Gagneja