Dicionários, JSON e Astronautas

Algum tempo atrás, fiquei sabendo que o endereço http://api.open-notify.org/ disponibilizava uma API bem curiosa: uma listagem, em JSON, da atual tripulação em todas (as duas) estações espaciais em redor da Terra. o resultado final da requisição é, no momento em que estou escrevendo esse texto, o seguinte:

```
{
"people": [
    {"craft": "ISS", "name": "Mark Vande Hei"},
    {"craft": "ISS", "name": "Oleg Novitskiy"},
    {"craft": "ISS", "name": "Pyotr Dubrov"},
    {"craft": "ISS", "name": "Thomas Pesquet"},
    {"craft": "ISS", "name": "Megan McArthur"},
    {"craft": "ISS", "name": "Shane Kimbrough"},
    {"craft": "ISS", "name": "Akihiko Hoshide"},
    {"craft": "Tiangong", "name": "Nie Haisheng"},
    {"craft": "Tiangong", "name": "Liu Boming"},
    {"craft": "Tiangong", "name": "Tang Hongbo"}
    ],
"number": 10,
"message": "success"
}
```

Ler um arquivo JSON é muito parecido com a leitura de um dicionário no Python. Na verdade, uma das primeiras coisas que fazemos ao trabalhar com JSON+Python é criar essa estrutura de dados. Um dicionário, de forma livre, nada mais é que um conjunto de dados classificados por pares chave-valor. Por exemplo, no caso dos dados dos astronautas acima, temos três chaves principais: *people*, *number* e *message*. A chave *message* possui um valor textual ("*success*"). A chave *number* possui um valor numérico (7). Já a chave people traz uma lista (observe os delimitadores "[" e "]"). Cada elemento da lista é um dicionário, com as chaves *craft* e *name*.

Esses dados são perfeitos para um primeiro exemplo de como usar o Python para processar o retorno de uma API WEB em JSON. Nesse exemplo, vamos:

1. Descobrir quais estações espaciais estão em órbita;
2. Contar quantos astronautas estão em cada estação espacial.

Nota: você precisa ter o módulo requests instalado. Se não tem, instale-o com o pip:

```
pip install requests
```

Para isso, a primeira coisa que precisamos é trazer os dados. Para isso, fazemos o seguinte:

```
import requests
retorno = requests.get("http://api.open-notify.org/astros.json")
```

Ele vai trazer um objeto do tipo *requests.models.Response*. E, em algum lugar dentro dele, tem o resultado da *requisição* (ou seja, o resultado da consulta feita ao site *open-notify*). A boa notícia é que o temos uma função pronta que já pega esses dados e transforma em um dicionário para a gente:

```
traduzido = retorno.json()
print(traduzido)
```

O resultado do *print* é:

```
{'people': [{'craft': 'ISS', 'name': 'Mark Vande Hei'}, {'craft': 'ISS',
'name': 'Oleg Novitskiy'}, {'craft': 'ISS', 'name': 'Pyotr Dubrov'},
{'craft': 'ISS', 'name': 'Thomas Pesquet'}, {'craft': 'ISS', 'name':
'Megan McArthur'}, {'craft': 'ISS', 'name': 'Shane Kimbrough'},
{'craft': 'ISS', 'name': 'Akihiko Hoshide'},
{'craft': 'Tiangong', 'name': 'Nie Haisheng'},
{'craft': 'Tiangong', 'name': 'Liu Boming'},
{'craft': 'Tiangong', 'name': 'Tang Hongbo'}],
'number': 10, 'message': 'success'}
```

A partir de agora, temos um dicionário padrão do Pyhton com nossos dados. E tudo fica mais simples! As informações que precisamos para responder as duas perguntas estão na lista que é o valor da chave *people*. Para simplificar nossa vida, vamos criar uma variável apenas com os dados que precisamos:

```
dados = traduzido["people"]
print(dados)
```

Resultando em:

```
[{'craft': 'ISS', 'name': 'Mark Vande Hei'},
 {'craft': 'ISS', 'name': 'Oleg Novitskiy'},
 {'craft': 'ISS', 'name': 'Pyotr Dubrov'},
 {'craft': 'ISS', 'name': 'Thomas Pesquet'},
 {'craft': 'ISS', 'name': 'Megan McArthur'},
 {'craft': 'ISS', 'name': 'Shane Kimbrough'},
 {'craft': 'ISS', 'name': 'Akihiko Hoshide'},
 {'craft': 'Tiangong', 'name': 'Nie Haisheng'},
 {'craft': 'Tiangong', 'name': 'Liu Boming'},
 {'craft': 'Tiangong', 'name': 'Tang Hongbo'}]
```

Vamos então responder a primeira pergunta: quais as estações espaciais que estão com tripulação nesse momento?

Para isso, vamos usar uma estrutura chamada de *set*. O *set* é, em resumo, uma lista sem elementos reptidos. Ou seja, vamos criar uma lista com cada uma das estações espaciais existentes na variável *traduzido* e depois transformamos em um *set*.

Forma 1: através de um *for* simples

```
estacoes = [] #declara uma lista vazia
for d in dados:
    estacoes.append(d["craft"])
estacoes_unicas = set(estacoes)
print(estacoes_unicas)
```

Resultado:

```
{'ISS', 'Tiangong'}
```

Forma 2: através de *list comprehension*

O *list comprehension* é uma forma mais eficiente computacionalmente de se fazer operações como a do código anterior. O código a seguir gera exatamente o mesmo resultado:

```
estacoes_unicas = set([x["craft"] for x in dados])
```

Quantos astronautas em cada estação?

Para responder essa pergunta, vamos criar um dicionário onde as chaves serão os nomes das estações, e os valores o número de astronautas. Para poder mostrar mais algumas coisas de dicionários, estou solenimente ignorando os *sets* que acabamos de criar.

```
1 for d in dados:
2     if d["craft"] in tripulacao.keys(): #Se existe essa chave
3         tripulacao[d["craft"]] += 1
4     else: #senão, cria a chave no dicionário
5         tripulacao[d["craft"]] = 1
6 print(tripulacao)
```

Resultado:

```
1 {'ISS': 7, 'Tiangong': 3}
```

Como o código anterior funciona? Vamos lá:

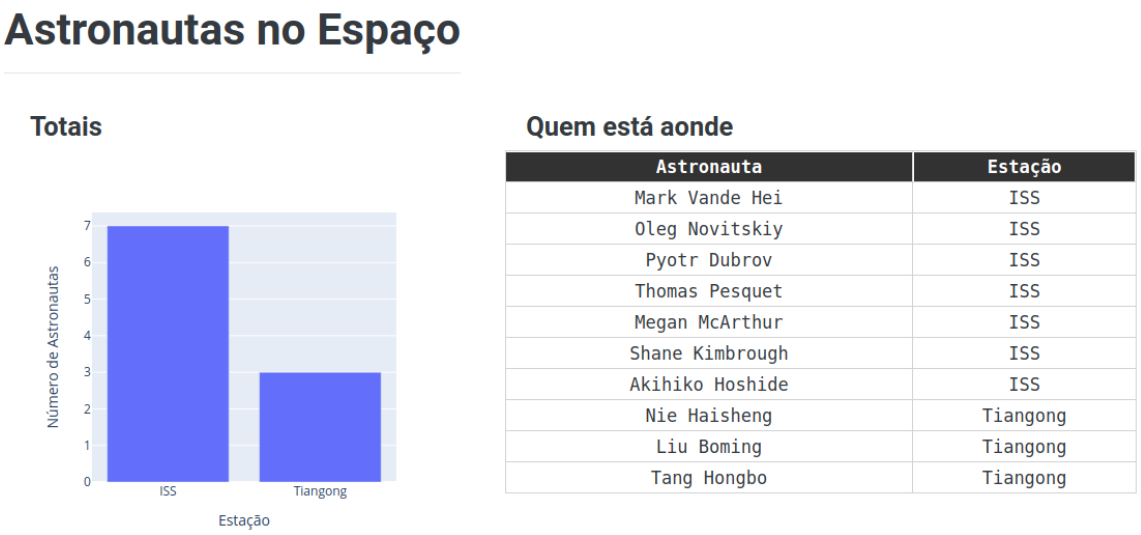
- A primeira coisa é tratar item a item da lista *dados*. Para isso, d vai iterar por toda ela.
- O método *keys()* do dicionário traz todas as chaves existentes. No caso, se a estação (indicada por d["craft"]) já existir em alguma chave de *tripulacao*, a linha 3 apenas soma um ao valor atual. Se não existir, a linha 5 cria esse conjunto *chave-valor*.

Em tempo: se nossa idéia for trabalhar com *dataframes*, é muito simples:

```
1 import pandas as pd
2 df = pd.DataFrame(dados)
```

Uma vez tendo os dados, podemos usar o Dash para criar um relatório - e até um Dashboard completo!

Focando no relatório, o resultado seria algo como o mostrado na Figura 1:



Astronauta	Estação
Mark Vande Hei	ISS
Oleg Novitskiy	ISS
Pyotr Dubrov	ISS
Thomas Pesquet	ISS
Megan McArthur	ISS
Shane Kimbrough	ISS
Akihiko Hoshide	ISS
Nie Haisheng	Tiangong
Liu Boming	Tiangong
Tang Hongbo	Tiangong

Figura 1: O resultado final do relatório

Já o código necessário para fazer esse relatório é o seguinte:

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_bootstrap_components as dbc
import dash_table
import plotly.express as px
import pandas as pd
import requests

#Obtém os dados
astros = requests.get("http://api.open-notify.org/astros.json")
tripulacao = astros.json()["people"]

#Transformamos os dados em um dicionário para depois transformar em DataFrame
#Note que estamos usando duas vezes list comprehension
dados = {"craft":[i["craft"] for i in tripulacao],
         "name":[i["name"] for i in tripulacao]}

df = pd.DataFrame(dados)
df.columns=["Estação", "Astronauta"]

#Cria o objeto com a tabela a ser mostrada
tabela = dash_table.DataTable(
    id='tabela',
    data=df.to_dict("records"),
    columns=[{"name": i, "id": i} for i in ["Astronauta", "Estação"]],
    style_cell={'textAlign': 'center'},
    style_header={
            'backgroundColor': 'rgb(50, 50, 50)',
            'color':'white',
            'fontWeight': 'bold',
            'textAlign': 'center'
        }
)

#Cria o objeto com o gráfico de barras
dados_sumarizados = pd.pivot_table(df, index=["Estação"], aggfunc=pd.Series.
    ↪ count).reset_index()
dados_sumarizados.columns = ["Estação", "Número de Astronautas"]
print(dados_sumarizados)
grafico = px.bar(dados_sumarizados,
    x="Estação",
    y="Número de Astronautas",
    height=400,
)

#Cria o objeto Dash
app = dash.Dash(external_stylesheets=[dbc.themes.LITERA])

#Cria a estrutura da página
app.layout = dbc.Container(
    [
        dbc.Row (
            html.Div([
                dcc.Store(id="store"),
                html.H1("Astronautas no Espaço"),
                html.Hr()
            ])
        ),
```

```
        dbc.Row ([
            dbc.Col (
                [
                dbc.Row ([
                    html.H4("Totais", style={"padding-left":20})
                ]),
                dbc.Row ([ html.Div([
                    dcc.Graph(
                        figure=grafico
                    )
                    ], id="div_grafico", style={"width": "100%"})
                ])
                ],
            width=5, style={"padding":20, "height":450}),
            dbc.Col (
                html.Div([
                dbc.Row ([
                    html.H4("Quem está aonde", style={"width": "75%", "padding-
                        ↪ left":20})
                ]),
                dbc.Row ([
                    html.Div([
                        tabela
                        ], id="div_bar", style={"width": "100%"})
                ])
                ], style={"padding":20, "height":450}),
            width=7
            )
        ], align="start", style={"border-style":"none", "border-color":"yellow"
            ↪ }
        )
    ], fluid=True )

#Executa o servidor
if __name__ == '__main__':
    app.run_server(debug=True)
```

A idéia desse texto foi mostrar um exemplo de como você pode ler arquivos via rede de dados que chegam em formato JSON e transformá-los em um relatório (e, por que não, extendê-lo posteriormente e transformá-lo em um dashboard). Esse padrão de comunicação está muito em voga, principalmente por sistemas REST e arquiteturas baseadas em microserviços. Como você pode ver, o Python tem um conjunto de bibliotecas que facilita muito a vida do programador.

Se você for trabalhar com dados em formato JSON, você vai usar bastante conceitos de listas e dicionários. Talvez seja interessante passar os olhos na documentação oficial do Python (https://docs.python.org/3/tutorial/datastructures.html)!

Em tempo: esse código está no site da disciplina! ;-)