

# Curso de Python do ZERO AO DS

com Meigarom do canal “Seja Um Data Scientist”

**Instagram:** @meigarom.datascience ( Mais informações sobre o Curso )

**LinkedIn:** <https://www.linkedin.com/in/meigarom/> ( Contato Profissional )

**Telegram:** <https://t.me/sejaumdatascientist> ( GRUPO DE ESTUDOS )

## Aula 04 - Estruturas de Controle

### Agenda:

1. Recapitulando.
2. Novas perguntas de negócio.
3. Planejamento da Solução.

**4. Estruturas de Dados -  
Listas**

**5. Estruturas de Controle -  
Condicionais**

**6. Estruturas de Controle -  
Laços**

## **1. Recapitulando**

**Aula 01: Começando com o python**

**Aula 02: Extração e Manipulação de Dados I**

**Aula 03: Transformação de Dados**

**Aula 04: Estruturas de Controle**

**1. Recado 01: Chegamos em 50% do curso  
Python do ZERO ao DS**

**2. Live ao Vivo na Sexta-feira, às 20h.  
Tema: Perguntas & Respostas sobre Ciência  
de Dados**

**+ Abertura da Comunidade DS**

### 3. É possível

## 2. Novas perguntas de negócio.

### 2.1. Recapitulando o desafio: ( <https://sejaumdatascientist.com/os-5-projetos-de-data-science-que-fara-o-recrutador-olhar-para-voce/> )

- **EMPRESA:** House Rocket
- **MODELO DE NEGÓCIO:** Compra casas com preço baixo e revendo com o preço mais alto.
- **QUAL O DESAFIO:** Encontrar bons negócios dentro do portfólio disponível, ou seja, encontrar casas com preço baixo, em ótima localização e que tenham um ótimo potencial de revenda por um preço mais alto.

### 2.2. Novas perguntas do CEO para você:

1. Qual a quantidade de imóveis por nível?
  - Nível 0 -> Preço entre R\$ 0 e R\$ 321.950
  - Nível 1 -> Preço entre R\$ 321.950 e R\$ 450.000
  - Nível 2 -> Preço entre R\$ 450.000 e

**R\$ 645.000**

**- Nível 3 -> Acima de R\$ 645.000**

**2. Adicione as seguintes informações ao imóvel:**

- O nome da Rua**
- O número do Imóvel**
- O nome do Bairro**
- O nome da Cidade**
- O nome da Estado**

**3. Adicionar o Nível dos imóveis no Mapa como uma Cor**

**4. Adicionar o Preço dos imóveis como o tamanho do ponto no mapa**

**5. Adicionar opções de filtros para eu fazer minhas próprias análises:**

- 1. Eu quero escolher visualizar imóveis com vista para água ou não.**
- 2. Eu quero filtrar os imóveis até um certo valor de preço.**

**6. Adicionar opções de filtros no último dashboard enviado:**

**1. Eu quero visualizar somente valor a partir de um data disponível para compra.**

## **3. Planejamento da solução:**

### **3.1. Produto Final ( O que eu vou entregar? Planilha, gráfico, modelo de ML, ... )**

- Email + 3 anexos:**
  - Email: As respostas das perguntas.**
    - Pergunta | Resposta**
  - Anexo 01: Um arquivo .csv com as novas informações requisitadas.**
  - Anexo 02: O mapa com os filtros requisitados.**
  - Anexo 03: O dashboard com os filtros requisitados.**

### **3.2. Ferramenta ( Qual ferramenta usar? )**

- Python 3.8.0**
- Jupyter Notebook**

### **3.3. Processo ( Como fazer? )**

**1. Qual a quantidade de imóveis por nível?**

- Nível 0 -> Preço entre R\$ 0 e R\$ 321.950
- Nível 1 -> Preço entre R\$ 321.950 e R\$ 450.000
- Nível 2 -> Preço entre R\$ 450.000 e R\$ 645.000
- Nível 3 -> Acima de R\$ 645.000

- Criar uma nova coluna vazia
- Popular essa nova colunas com os valor condicionados aos intervalos.

**2. Adicione as seguintes informações ao imóvel:**

- O nome da Rua
  - O número do Imóvel
  - O nome do Bairro
  - O nome da Cidade
  - O nome da Estado
- Onde eu vou encontrar essas informações?
  - Qual dado em tenho na base que vai me ajudar a “linkar” essas novas informações.

- Coletar os novos dados.
- Anexar os dados na base original.

### **3. Adicionar o Nível dos imóveis no Mapa como uma Cor**

- Adicionar a nova coluna “Nível” como uma cor no mapa.

### **4. Adicionar o Preço dos imóveis como o tamanho do ponto no mapa**

- Adicionar coluna “Price” como o tamanho dos pontos no mapa.

### **5. Adicionar opções de filtros para eu fazer minhas próprias análises:**

1. Eu quero escolher visualizar imóveis com vista para água ou não.
2. Eu quero filtrar os imóveis até um certo valor de preço.

- Descobrir uma biblioteca que adicionar filtros nos gráficos.
- Fazer com que esses filtros sejam “linkados” com o conjunto de dados.

### **6. Adicionar opções de filtros no último**

**dashboard enviado:**

**1. Eu quero visualizar somente valor a partir de um data disponível para compra.**

- **Descobrir uma biblioteca que adicionar filtros nos gráficos.**
- **Fazer com que esses filtros sejam “linkados” com o conjunto de dados.**

## **4. As estruturas de Dados em Python**

**- Listas**

- **Dicionários ( Mostrarei na próxima aula, Aula 03 )**
- **Tuples ( Mostrar no avançado quando eu ensinar SQL + Python )**
- **Dataframes ( Mostrarei na próxima aula, Aula 03 )**

### **4.1. Listas:**

- **Possui valores, tanto numéricos quanto categóricos**
- **Os valores são mapeados por uma**



posição na lista, ou seja, cada valor possui um index.

- As listas precisam de um nome

## Dentro do Jupyter Notebook

```
# =====
```

```
# Estrutura de Dados - Lista
```

```
# =====
```

```
# Lista de números
```

```
a1 = [2, 3, 5, 6, 3, 65, 4]
```

```
# Lista de Strings
```

```
a2 = ['seja', 'um', 'data', 'scientist']
```

```
# Lista de Strings + Números
```

```
a3 = ['seja', 1, 'data', 10, 'scientist', 100 ]
```

```
# Lista dentro de lista
```

```
a4 = ['seja', [1, 2, 3], 'data', 10, 'scientist', 100]
```

```
# Como acessar os valores da lista?
```

```
# Através dos index
```

```
print( a3[5] )
```

```
print( a3[0:4] )
```

```
print( a4[1] )  
print( a4[1][0] )
```

```
# Como adicionar novos elementos na lista?  
a4.append( 50 ) # -> adiciona no final da list  
a4.insert( 0, 50 ) # -> ( posicao, valor )
```

```
print( a4 )
```

```
# Como medir o tamanho da list?  
#len( a3 )
```

```
# Como criar uma lista vazia  
# a = []
```

## 5. Estrutura de Controle - Condicionais

- Condicionais:
  - Seleção de colunas e filtragem de linhas condicionados ( baseado em condições )

# Dentro do Jupyter Notebook

```
# =====  
# Estrutura de Controle - Condicionais  
# =====  
# Igual  
# Maior  
# Maior ou igual que  
# Menor  
# Menor ou igual  
  
df = pd.read_csv( 'kc_house_data.csv' )  
  
print( df[(df['bedrooms'] == 83) ] )  
  
# Logica E ( AND ) - &  
# Logica da multiplicacao  
# bedrooms | waterfront | resultado  
# True | True | True  
# False | True | False  
# True | False | False  
# False | False | False  
  
print( df[(df['bedrooms'] == 83) &
```

```
(df['waterfront'] == 2) ].head(), end='\n\n' )
```

```
# Logica OU ( OR ) - I
```

```
# Logica da soma
```

```
# bedrooms I waterfront I resultado
```

```
# True I True I True
```

```
# False I True I True
```

```
# True I False I True
```

```
# False I False I False
```

```
print( df[(df['bedrooms'] == 83) I
```

```
(df['waterfront'] == 1) ].head() )
```

```
#
```

```
# =====
```

```
# Estrutura de Controle - Condicionais
```

```
# =====
```

```
df = pd.read_csv( 'kc_house_data.csv' )
```

```
# Consigo abordar apenas os casos
```

```
verdadeiros
```

```
df.loc[(df['bedrooms'] == 3) & (df['waterfront']
```

```
== 1), 'teste'] = 10
```

```
# Como eu faço para abordar os casos Falsos
```

também?

## 6. Estrutura de Controle - O Laço

- **Laços:**
  - São repetições de comandos até que uma condição seja satisfeita
  
- **Laço FOR:**
  - Aplica transformações de dados em intervalos escolhidos pelo programador.
  - É necessário conhecer o tamanho do laço. ( Até quando vou repetir os comandos )
  
- **Laço WHILE:**
  - Mantem o fluxo de código continuamente.
  - Usando quando o tamanho do laço é desconhecido ( Eu não até quando a repetição vai ocorrer )

## Dentro do Jupyter Notebook

## **# Exemplo do laço FOR**

```
if ( df.loc[0, 'bedrooms'] == 3 ) & ( df.loc[0,  
'waterfront'] == 1 ):  
    df.loc[0, 'teste'] = 10
```

```
else:  
    df.loc[0, 'teste'] = 5
```

## **# Como comparar todas as linhas?**

```
for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:  
    print( i )
```

## **# Usando uma função para comparar todas as linhas**

```
for i in range( 0, len( df ) ):  
    print( i )
```

## **# Usando uma função para criar a lista**

```
for i in range( 0, len( df ) ):  
    if ( df.loc[i, 'bedrooms'] == 0 ) & ( df.loc[i,  
'waterfront'] == 1 ):  
        df.loc[i, 'teste'] = 10
```

```
else:
```

```
df.loc[i, 'teste'] = 5
```

## **# Exemplo do laço WHILE**

```
import requests as r  
i = 1  
dataset = pd.DataFrame()  
while True:  
    print( 'page: {}'.format( i ) )  
    #url = 'https://jobs.github.com/  
positions.json?  
description=python&full_time=true&location=  
new+york&page={}'.format( i )  
    url = 'https://jobs.github.com/  
positions.json?  
full_time=true&page={}'.format( i )  
    response = r.request( 'GET', url )  
  
    if response.json() != []:  
        data = response.json()[0]  
        df = pd.DataFrame( data, index=[0] )  
        print( df[['id', 'type']] )  
        dataset = pd.concat( [dataset, df], axis=0 )  
        i = i + 1
```

**else:**  
**break**

## **7. Executando o PROCESSO planejado:**

**1. Qual a quantidade de imóveis por nível?**

- **Nível 0 -> Preço entre R\$ 0 e R\$ 321.950**
- **Nível 1 -> Preço entre R\$ 321.950 e R\$ 450.000**
- **Nível 2 -> Preço entre R\$ 450.000 e R\$ 645.000**
- **Nível 3 -> Acima de R\$ 645.000**

- **Criar uma nova coluna vazia**
- **Popular essa nova colunas com os valor condicionados aos intervalos.**

## **Dentro do Jupyter Notebook**



**# Exemplo da Aplicação 01: Definir os níveis de preços**

**# 0 até 321.950 = Level 0**

**# Entre 321.950 e 450.000 = Level 1**

**# Entre 450.000 e 645.000 = Level 2**

**# Acima de 645.000 = Level 3**

**# define level of prices**

**for i in range( len( df ) ):**

**if df.loc[i, 'price'] <= 321950:**

**df.loc[i, 'level'] = 0**

**elif ( df.loc[i,'price'] > 321950 ) &  
( df.loc[i,'price'] <= 450000 ):**

**df.loc[i, 'level'] = 1**

**elif ( df.loc[i,'price'] > 450000 ) &  
( df.loc[i,'price'] <= 645000 ):**

**df.loc[i, 'level'] = 2**

**else:**

**df.loc[i, 'level'] = 3**

**df[['id', 'level']].groupby( 'level' ).shape**

=====

=====

## 2. Adicione as seguintes informações ao imóvel:

- O nome da Rua
  - O número do Imóvel
  - O nome do Bairro
  - O nome da Cidade
  - O nome da Estado
- Onde eu vou encontrar essas informações?
- Qual dados em tenho na base que vai me ajudar a “linkar” essas novas informações.
  - Coletar os novos dados.
  - Anexar os dados na base original.

## Dentro do Jupyter Notebook

```
from geopy.geocoders import Nominatim
```

```
# initialize Nominatim API  
geolocator =
```

```
Nominatim(user_agent="geoapiExercises")
response = geolocator.reverse( '47.5112,
-122.257' )
```

```
print( response.raw['address']['road'] )
print( response.raw['address']
['house_number'] )
print( response.raw['address']
['neighbourhood'] )
print( response.raw['address']['city'] )
print( response.raw['address']['county'] )
print( response.raw['address']['state'] )
```

```
# Read Data
```

```
data = pd.read_csv( 'kc_house_data.csv' )
data = data.head()
```

```
# Create empty rows
```

```
data['road'] = 'NA'
data['house_number'] = 'NA'
data['city'] = 'NA'
data['county'] = 'NA'
data['state'] = 'NA'
```

```
for i in range( len( data ) ):
    print( 'Loop: {}/{}'.format( i, len( data ) ) )
```

```
# make request
query = str( data.loc[i, 'lat'] ) + ',' +
str( data.loc[i, 'long'] )
response = geolocator.reverse( query )

# parse data
if 'house_number' in
response.raw['address']:
    data.loc[i, 'house_number'] =
response.raw['address']['house_number']

if 'road' in response.raw['address']:
    data.loc[i, 'road'] =
response.raw['address']['road']

if 'city' in response.raw['address']:
    data.loc[i, 'city'] = response.raw['address']
['city']

if 'county' in response.raw['address']:
    data.loc[i, 'county'] =
response.raw['address']['county']

if 'state' in response.raw['address']:
    data.loc[i, 'state'] =
response.raw['address']['state']
```



### **3. Adicionar o Nível dos imóveis no Mapa como uma Cor**

**- Adicionar a nova coluna “Nível” como uma cor no mapa.**

### **4. Adicionar o Preço dos imóveis como o tamanho do ponto no mapa**

**- Adicionar coluna “Price” como o tamanho dos pontos no mapa.**

**Dentro do Jupyter Notebook**

```
data = pd.read_csv( 'kc_house_data.csv' )  
houses = data[['id', 'lat', 'long', 'price']].copy()
```

```
# define level of prices  
for i in range( len( houses ) ):  
    if houses.loc[i, 'price'] <= 321950:  
        houses.loc[i, 'level'] = 0
```

```
elif ( houses.loc[i,'price'] > 321950 ) &  
( houses.loc[i,'price'] <= 450000 ):  
    houses.loc[i, 'level'] = 1
```

```
elif ( houses.loc[i,'price'] > 450000 ) &  
( houses.loc[i,'price'] <= 645000 ):  
    houses.loc[i, 'level'] = 2
```

```
else:  
    houses.loc[i, 'level'] = 3
```

```
houses['level'] = houses['level'].astype( int )
```

```
fig = px.scatter_mapbox( houses,  
                        lat="lat",  
                        lon="lon",  
                        color="level",  
                        size="price",
```

```
color_continuous_scale=px.colors.cyclical.lc  
Fire,
```

```
size_max=15,  
zoom=10)
```

```
fig.update_layout(mapbox_style="open-street-  
map")
```

```
fig.update_layout(height=600,  
margin={"r":0,"t":0,"l":0,"b":0})  
fig.show()
```

=====

=====

**5. Adicionar opções de filtros para eu fazer minhas próprias análises:**

- 1. Eu quero escolher visualizar imóveis com vista para água ou não.**
- 2. Eu quero filtrar os imóveis até um certo valor de preço.**

- Descobrir uma biblioteca que adicionar filtros nos gráficos.**
- Fazer com que esses filtros sejam “linkados” com o conjunto de dados.**

=====

=====

## **Dentro do Jupyter Notebook**

```

import ipywidgets as widgets
from ipywidgets import Layout

from ipywidgets import fixed

# Prepare Data
df = pd.read_csv( 'kc_house_data.csv' )
df['is_waterfront'] =
df['waterfront'].apply( lambda x: 'yes' if x == 1
else 'no' )

# define level of prices
df['level'] = df['price'].apply( lambda x: 0 if x <=
321950 else
                                1 if ( x > 321950 ) &
( x <= 450000) else
                                2 if ( x > 450000 ) &
( x <= 645000 ) else 3 )
df['level'] = df['level'].astype( int )
style = {'description_width': 'initial'}

# Interactive buttons
price_limit = widgets.IntSlider(
    value = 540000,
    min = 75000,
    max = 7700000,

```



```
    step = 1,  
    description='Maximun Price',  
    disable=False,  
    style=style  
)
```

```
waterfront_bar = widgets.Dropdown(  
    options = df['is_waterfront'].unique().tolist(),  
    value = 'yes',  
    description = 'Water View',  
    disable=False  
)
```

```
def update_map( df, waterfront, limit ):  
    houses = df[(df['price'] <= limit) &  
                (df['is_waterfront'] == waterfront)]  
    [['id', 'lat', 'long', 'price', 'level', 'bathrooms']]
```

```
fig = px.scatter_mapbox( houses,  
                          lat="lat",  
                          lon="long",  
                          size="price",  
                          #color="bathrooms",  
                          color="level",
```

```
color_continuous_scale=px.colors.cyclical.Ice  
Fire,
```

```
size_max=15,  
zoom=10)
```

```
fig.update_layout(mapbox_style="open-  
street-map")
```

```
fig.update_layout(height=600,  
margin={"r":0,"t":0,"l":0,"b":0})
```

```
fig.show()
```

```
widgets.interactive( update_map,  
df=fixed( df ), waterfront=waterfront_bar,  
limit=price_limit )
```

```
=====
```

```
=====
```

**6. Adicionar opções de filtros no último dashboard enviado:**

**1. Eu quero visualizar somente valor a partir de um data disponível para compra.**

- Descobrir uma biblioteca que adicionar filtros nos gráficos.
- Fazer com que esses filtros sejam “linkados” com o conjunto de dados.

=====

=====

## Dentro do Jupyter Notebook

```
import ipywidgets as widgets
from matplotlib import gridspec
from matplotlib import pyplot as plt
```

```
# Prepare dataset
```

```
df = pd.read_csv( 'kc_house_data.csv' )
```

```
# Change data format
```

```
df['year'] =
```

```
pd.to_datetime( df['date'] ).dt.strftime( '%Y' )
```

```
df['date'] =
```

```
pd.to_datetime( df['date'] ).dt.strftime( '%Y-
%m-%d' )
```

```
df['year_week'] =
```

```
pd.to_datetime( df['date'] ).dt.strftime( '%Y-
```

```
%U' )
```

```
# Widget to control data
```

```
date_limit = widgets.SelectionSlider(  
    options =  
df['date'].sort_values().unique().tolist(),  
    value='2014-12-01',  
    description='Disponível',  
    disable=False,  
    continuous_update=False,  
    orientation='horizontal',  
    readout=True  
)
```

```
def update_map( data, limit ):
```

```
    # Filter data
```

```
    df = data[data['date'] >= limit].copy()
```

```
    fig = plt.figure( figsize=(24, 12) )
```

```
    specs = gridspec.GridSpec( ncols=2,  
nrows=2, figure=fig )
```

```
    ax1 = fig.add_subplot( specs[0, :] ) # First  
Row
```

```
    ax2 = fig.add_subplot( specs[1, 0] ) # First
```

**Row First Column**

```
ax3 = fig.add_subplot( specs[1, 1] ) #
```

**Second Row First Column**

```
by_year = df[['id',  
'year']].groupby( 'year' ).sum().reset_index()  
ax1.bar( by_year['year'], by_year['id'] )
```

**# Frist Graph**

```
by_day = df[['id',  
'date']].groupby( 'date' ).mean().reset_index()  
ax2.plot( by_day['date'], by_day['id'] )  
ax2.set_title( "Title: Avg Price by Day" )
```

```
df['year_week'] =  
pd.to_datetime( df['date'] ).dt.strftime( '%Y-  
%U' )
```

```
by_week_of_year = df[['id',  
'year_week']].groupby( 'year_week' ).mean().re  
set_index()
```

```
ax3.bar( by_week_of_year['year_week'],  
by_week_of_year['id'] )  
ax3.set_title( "Title: Avg Price by Week of  
Year" )  
plt.xticks( rotation=60);
```

```
widgets.interactive( update_map,  
data=fixed( df ), limit=date_limit )
```

```
=====
```

```
=====
```

## 7. Exercícios:

**Novas perguntas do CEO para você:**

**1. Qual a média do preço de compra dos imóveis por “Nível”?**

- Nível 0 -> Preço entre R\$ 0 e R\$ 321.950
- Nível 1 -> Preço entre R\$ 321.950 e R\$ 450.000
- Nível 2 -> Preço entre R\$ 450.000 e R\$ 645.000
- Nível 3 -> Acima de R\$ 645.000

**2. Qual a média do tamanho da sala de estar dos imóveis por “Size” ?**

- Size 0 -> Tamanho entre 0 e 1427 sqft
- Size 1 -> Tamanho entre 1427 e

**1910 sqft**

**- Size 2 -> Tamanho entre 1910 e**

**2550 sqft**

**- Size 3 -> Tamanho acima de 2550**

**sqft**

**3. Adicione as seguinte informações ao conjunto de dados original:**

**- Place ID: Identificação da localização**

**- OSM Type: Open Street Map type**

**- Country: Nome do País**

**- Country Code: Código do País**

**4. Adicione os seguinte filtros no Mapa:**

**- Tamanho mínimo da área da sala de estar.**

**- Número mínimo de banheiros.**

**- Valor Máximo do Preço.**

**- Tamanho máximo da área do porão.**

**- Filtro das Condições do Imóvel.**

**- Filtro por Ano de Construção.**

**5. Adicione os seguinte filtros no Dashboard:**

- Filtro por data disponível para compra.**
- Filtro por ano de renovação.**
- Filtro se possui vista para a água ou não.**