# aula04_estruturas_controle

February 28, 2021

## 1 Listas

[69]:
```
# Estruturas de Dados - Lista
#a = [ 2, 3, 4, 5, 6, 7]   -> Valores
#    [ 0, 1, 2, 3, 4, 5]   -> Posicoes

# Lista de Numeros
a = [ 2, 3, 4, 5, 6, 7]

# Lista de String
b = ['seja', 'um', 'cientista', 'de', 'dados']

# Lista de String + numeros
c = ['seja', 1, 'cientista', 10, [ 10, 'dua', 32, 'dfa', 'sfa'] ]

# Como adicionar um novo elemento na lista
c.append( 50 ) # Sempre coloca o elemento no final da lista

c.insert( 4, 40 ) # Sempre colocar o elemento na posicao desejada ( posicao,␣
 ↪elemento )

# Como medir o tamanho da lista
print( c )
len( c )

# Criar uma lista vazia
d = []
```

```
['seja', 1, 'cientista', 10, 40, [10, 'dua', 32, 'dfa', 'sfa'], 50]
```

[70]:
```
d
```

[70]:
```
[]
```

## 2 Estrutura de controle - Condicional

[ ]:
```
# Condicionais que nos temos:
# Igual
# Maior
# Maior ou igual
# Menor
# Menor ou igual
```

[1]:
```
import pandas as pd
```

```
/Users/meigarom.lopes/.pyenv/versions/3.8.0/envs/pythonzeroaods/lib/python3.8/site-packages/pandas/compat/__init__.py:97: UserWarning: Could not import the lzma module. Your installed Python is incomplete. Attempting to use lzma compression will result in a RuntimeError.
  warnings.warn(msg)
```

[2]:
```
data = pd.read_csv( 'kc_house_data.csv' )
```

[14]:
```
# Logicas E ( AND )
# Logica de Multiplicacao

# Bedrooms | Floors
#    TRUE  |  TRUE -> Resultado
#    TRUE  |  FALSE -> ERRO
#    FALSE |  TRUE -> ERRO
#    FALSE |  FALSE -> ERRO

# TRUE & TRUE
#df = data[(data['bedrooms'] == 4) & ( data['floors'] == 2 )]

# TRUE & FALSE
#df = data[(data['bedrooms'] == 4) & ( data['floors'] == 20 )]

# FALSE & TRUE
#df = data[(data['bedrooms'] == 50) & ( data['floors'] == 2 )]

# FALSE & FALSE
df = data[(data['bedrooms'] == 50) & ( data['floors'] == 20 )]
df.head()
```

[14]:
```
Empty DataFrame
Columns: [id, date, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, waterfront, view, condition, grade, sqft_above, sqft_basement, yr_built, yr_renovated, zipcode, lat, long, sqft_living15, sqft_lot15]
Index: []

[0 rows x 21 columns]
```

[ ]:

[20]:
```
# Logicas OU ( OR )
# Logica de Soma

# Bedrooms | Floors
#    TRUE  |  TRUE -> Resultado
#    TRUE  |  FALSE -> Resultado
#    FALSE |  TRUE -> Resultado
#    FALSE |  FALSE -> ERRO

# TRUE & TRUE
#df = data[(data['bedrooms'] == 4) | ( data['floors'] == 2 )]

# TRUE & FALSE
#df = data[(data['bedrooms'] == 4) | ( data['floors'] == 20 )]

# FALSE & TRUE
#df = data[(data['bedrooms'] == 50) | ( data['floors'] == 2 )]

# FALSE & FALSE
#df = data[(data['bedrooms'] == 50) | ( data['floors'] == 20 )]
#df.head()
```

[ ]:
```
#1. Qual a quantidade de imóveis por nível?
#        - Nivel 0: Preço entre R$ 0.00 e R$ 321.950
#        - Nivel 1: Preço entre R$ 321.950 e R$ 450.000
#        - Nivel 2: Preço entre R$ 450.000 e R$ 645.000
#        - Nivel 3: Preço acima de R$ 645.000

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

data['nivel'] = NA
data.loc[data[(data['price'] >= 0) & (data['price'] < 321950)], 'nivel'] =␣
 ↪'nivel_0'
data.loc[data[(data['price'] >= 321950) & (data['price'] < 450000)], 'nivel'] =␣
 ↪'nivel_1'
data.loc[data[(data['price'] >= 450000) & (data['price'] < 645000)], 'nivel'] =␣
 ↪'nivel_2'
data.loc[data[(data['price'] >= 645000)], 'nivel'] = 'nivel_3'
```

## 3 Laco FOR

[60]:
```
data = pd.read_csv( 'kc_house_data.csv' )
```

[59]:
```
# LOOP FOR
for i in range( 0, len( data ) ):
    if (data.loc[i, 'price'] > 0) & ( data.loc[i, 'price'] < 321950):
        data.loc[i, 'nivel'] = 'nivel_0'

    elif (data.loc[i, 'price'] >= 321950) & ( data.loc[i, 'price'] < 450000):
        data.loc[i, 'nivel'] = 'nivel_1'

    elif (data.loc[i, 'price'] >= 450000) & ( data.loc[i, 'price'] < 645000):
        data.loc[i, 'nivel'] = 'nivel_2'

    else:
        data.loc[i, 'nivel'] = 'nivel_3'
```

[62]:
```
data.head()
```

[62]:

| | id | date | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront | view | … | sqft_above | sqft_basement | yr_built | yr_renovated | zipcode | lat | long | sqft_living15 | sqft_lot15 | nivel |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7129300520 | 20141013T000000 | 221900.0 | 3 | 1.00 | 1180 | 5650 | 1.0 | 0 | 0 | … | 1180 | 0 | 1955 | 0 | 98178 | 47.5112 | -122.257 | 1340 | 5650 | nivel_0 |
| 1 | 6414100192 | 20141209T000000 | 538000.0 | 3 | 2.25 | 2570 | 7242 | 2.0 | 0 | 0 | … | 2170 | 400 | 1951 | 1991 | 98125 | 47.7210 | -122.319 | 1690 | 7639 | NaN |
| 2 | 5631500400 | 20150225T000000 | 180000.0 | 2 | 1.00 | 770 | 10000 | 1.0 | 0 | 0 | … | 770 | 0 | 1933 | 0 | 98028 | 47.7379 | -122.233 | 2720 | 8062 | NaN |
| 3 | 2487200875 | 20141209T000000 | 604000.0 | 4 | 3.00 | 1960 | 5000 | 1.0 | 0 | 0 | … | 1050 | 910 | 1965 | 0 | 98136 | 47.5208 | -122.393 | 1360 | 5000 | NaN |
| 4 | 1954400510 | 20150218T000000 | 510000.0 | 3 | 2.00 | 1680 | 8080 | 1.0 | 0 | 0 | … | 1680 | 0 | 1987 | 0 | 98074 | 47.6168 | -122.045 | 1800 | 7503 | NaN |

[5 rows x 22 columns]

[ ]:
```

```

[ ]:

## 4 Laco WHILE

[83]:
```
import requests as r


i = 1
dataset = pd.DataFrame()
while True:
    print( 'page:{}'.format( i ) )
    url = 'https://jobs.github.com/positions.json?page={}'.format( i )
    response = r.request( 'GET', url )

    if response.json() != []:
        data = response.json()[0]
        df = pd.DataFrame( data, index=[0] )

        dataset = pd.concat( [ dataset, df], axis=0 )
        i = i + 1

    else:
        break
```

```
page:1
page:2
page:3
page:4
page:5
```

[ ]:

## 5 Respondendo as perguntas do CEO

[ ]:
```
#        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 tem essas informações? API chamada GEOPY
#                                - Tem no banco de dados da empresa? NO
#                                - Tem essas informações são fornecidas por um␣
 ↪API. SIM
```

```
#                                 - Dentro de arquivo na pasta do meu colega de trabalho chamado Legilson NAO
#
#                 - Qual dados eu tenho na minha base que eu consiga fazer o link.
#                 - Como coletar esse dado e como anexá-lo no conjunto de dados original.
```

[84]:
```
data = pd.read_csv( 'kc_house_data.csv' )
```

[89]:
```
from geopy.geocoders import Nominatim
```

[92]:
```
# 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'] )
```

[ ]:
```

```

[ ]:
```

```

## 6 Filtro iterativos no map

[124]:
```
import plotly.express as px
```

[126]:
```
data = pd.read_csv( 'kc_house_data.csv' )
houses = data[['id', 'lat', 'long', 'price']].copy()

# define level
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='long',
                         color='level',
                         size='price',
                         color_continuous_scale=px.colors.cyclical.IceFire,
                         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()
```

## 7 Adicionando Filtros iterativos

[141]:
```
import ipywidgets as widgets
from ipywidgets import fixed
```

[143]:
```
df = pd.read_csv( 'kc_house_data.csv' )

df['is_waterfront'] = df['waterfront'].apply( lambda x: 'yes' if x == 1 else␣
 ↪'no' )

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'}

# Iterative buttons
price_limit = widgets.IntSlider(
    value = 540000,
    min = 75000,
    max = 77000000,
    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']]

    fig = px.scatter_mapbox( houses,
                             lat='lat',
                             lon='long',
                             color='level',
                             size='price',
                             color_continuous_scale=px.colors.cyclical.IceFire,
                             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()
```

[144]:
```
widgets.interactive( update_map, df=fixed( df ), waterfront=waterfront_bar,␣
 ↪limit=price_limit)
```

```
interactive(children=(Dropdown(description='Water View', index=1, options=('no',␣
 ↪'yes'), value='yes'), IntSlid…
```

## 8 Iteravidade com o dashboard

[145]:
```
import ipywidgets as widgets
from matplotlib import gridspec
from matplotlib import pyplot as plt
```

[156]:
```
# prepare dataset
data = pd.read_csv( 'kc_house_data.csv' )

# change date format
data['year'] = pd.to_datetime( data['date'] ).dt.strftime( '%Y' )
```

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

# Widgets to control data
date_limit = widgets.SelectionSlider(
    options=df['date'].sort_values().unique().tolist(),
    value = '2014-12-01',
    description = 'Disponivel',
    continuous_update=False,
    orientation='horizontal',
    readout=True )


def update_map( data, limit ):
    # Filter data
    df = data[data['date'] >= limit ].copy()

    fig = plt.figure( figsize=(21,12) )
    specs = gridspec.GridSpec( ncols=2, nrows=2, figure=fig )

    ax1 = fig.add_subplot( specs[0, :] ) # First rows
    ax2 = fig.add_subplot( specs[1, 0] ) # Second Row First Column
    ax3 = fig.add_subplot( specs[1, 1] ) # Second Row Second Column  rows

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

    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' )

    by_week_of_year = df[['id', 'year_week']].groupby( 'year_week').mean().
 ↪reset_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 );
```

[157]:
```
widgets.interactive( update_map, data=fixed( data ), limit=date_limit)
```

```
interactive(children=(SelectionSlider(continuous_update=False,␣
 ↪description='Disponivel', index=212, options=('…
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: