Construção do histograma da distribuição:

```
import matplotlib.pyplot as plt

# Construção do histograma
amp = data.max()-data.min()
eixo_x = list(range(data.min(), data.max()+1, 5))
classes = pd.interval_range(start=data.min()-0.5, end=data.max()+0.5, freq=1)
histograma = data.hist(bins=[classe.left for classe in classes]+[data.max()
+0.5],
                       color='blue', edgecolor='black', grid=False)

# Inserção de atributos ao gráfico
histograma.set(xlabel='Idade', ylabel='Frequência',
               title='Distribuição de frequências das idades em uma turma de
               alunos',
               xticks=eixo_x)
plt.show()
```

- **amp = data.max()-data.min()**: Isso calcula a amplitude dos dados de idade. A amplitude é a diferença entre o valor máximo (**data.max()**) e o valor mínimo (**data.min()**) nos dados de idade. Ela é usada para determinar a faixa de valores que serão divididos em intervalos de classe.
- **eixo_x = list(range(data.min(), data.max()+1, 5))**: Aqui, **eixo_x** é uma lista de valores inteiros que representam as marcações no eixo x do histograma. Os valores são gerados usando a função **range()** e começam no valor mínimo dos dados (**data.min()**) até o valor máximo dos dados (**data.max()**) mais 1, com um passo de 5. Isso cria uma lista de marcações a cada 5 unidades no eixo x.
- **classes = pd.interval_range(start=data.min()-0.5, end=data.max()+0.5, freq=1)**: Esta linha cria uma série de intervalos de classe usando o **pandas**. Os intervalos de classe são gerados com base nos valores mínimo e máximo dos dados de idade. O argumento **start** é definido como **data.min() - 0.5**, e o argumento **end** é definido como **data.max() + 0.5**, com uma frequência (**freq**) de 1 unidade. Isso cria intervalos de classe que cobrem toda a faixa de idades com um espaçamento de 1 unidade.
- **histograma = data.hist(bins=[classe.left for classe in classes]+[data.max()+0.5], color='blue', edgecolor='black', grid=False)**: Aqui, o código cria o histograma propriamente dito:
- **data.hist()**: Esta função cria o histograma dos dados de idade contidos em **data**.
- **bins=[classe.left for classe in classes]+[data.max()+0.5]**: O argumento **bins** define os intervalos de classe para o histograma. Ele usa a lista de intervalos de classe **classes** e adiciona um intervalo adicional com limite superior igual ao valor máximo dos dados mais 0.5. Isso garante que o último intervalo inclua o valor máximo dos dados.
- **color='blue'**: Define a cor das barras do histograma como azul.
- **edgecolor='black'**: Define a cor das bordas das barras como preta.
- **grid=False**: Desativa as grades de fundo no gráfico.

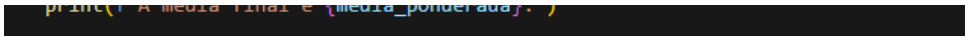
- **histograma.set(xlabel='Idade', ylabel='Frequência', title='Distribuição de frequências das idades em uma turma de alunos', xticks=eixo_x)**: Esta parte adiciona atributos ao gráfico:
- **xlabel='Idade'**: Define o rótulo do eixo x como "Idade".
- **ylabel='Frequência'**: Define o rótulo do eixo y como "Frequência".
- **title='Distribuição de frequências das idades em uma turma de alunos'**: Define o título do gráfico.
- **xticks=eixo_x**: Define as marcações no eixo x com base na lista **eixo_x** que você criou anteriormente.

Medidas de tendência central da distribuição

Mean() = média
Median() = mediana
Mode() = moda

```
# Cálculo das medidas de tendência central
media = data.mean()
print(f'A média da distribuição de idades é {media}.')
mediana = data.median()
print(f'A mediana da distribuição de idades é {mediana}.')
moda = list(data.mode())# mode retorna uma série de dados!
print(f'A moda da distribuição de idades é {moda[0]}.')
```

Média ponderada de uma série de dados



- **Criação do conjunto de dados**:
- notas = pd.DataFrame({'Notas':[86, 96, 82, 98, 100],
  'Pesos':[0.5, 0.15, 0.2, 0.1, 0.05]})
- Neste trecho, um DataFrame chamado **notas** é criado usando a biblioteca **pandas**. O DataFrame possui duas colunas: 'Notas', que armazena as notas atribuídas aos diferentes componentes, e 'Pesos', que armazena os pesos associados a cada componente. O conjunto de dados contém informações sobre a média do teste, prova bimestral, prova final, informática e dever de casa.
- **Definição dos índices do DataFrame**:
- notas.index = ['Média do teste', 'Prova bimestral', 'Prova final',
  'Informática', 'Dever de casa']

- Esta linha define os índices (rótulos das linhas) do DataFrame **notas**. Cada índice corresponde a um componente de avaliação.
- **Cálculo da Média Ponderada usando a definição**:

- media_ponderada = sum(notas['Notas'] * notas['Pesos']) / notas['Pesos'].sum()

- Nesta linha, o código calcula a média ponderada manualmente. Ele multiplica as notas pelo peso de cada componente, soma esses produtos e divide pelo somatório dos pesos. O resultado é armazenado na variável **media_ponderada**.
- **Exibição da Média Ponderada calculada usando a definição**:
- print(f'A média final é {media_ponderada}.')

- O resultado da média ponderada calculada usando a definição é exibido na tela.
- **Cálculo da Média Ponderada usando o NumPy**:
- import numpy as np
  media_ponderada = np.average(a=notas['Notas'], weights=notas['Pesos'])

- Aqui, o código utiliza a função **np.average()** do NumPy para calcular a média ponderada. Ela aceita um array de valores (**a**) e um array de pesos (**weights**). O resultado é armazenado novamente na variável **media_ponderada**.
- **Exibição da Média Ponderada calculada usando o NumPy**:
- print(f'A média final é {media_ponderada}.')

- O resultado da média ponderada calculada usando o NumPy é exibido na tela.

Média ponderada por agrupamento de dados

```
print(f'Preço médio do {medias.index[indice]}: {medias[indice]:.2f}')
```

- **Definição da Função media_ponderada**:
- def media_ponderada(dataframe, col_valores, col_pesos):
  valores = dataframe[col_valores]
  pesos = dataframe[col_pesos]
  return (valores * pesos).sum() / pesos.sum()

- Esta função **media_ponderada** aceita três argumentos:
- **dataframe**: O DataFrame contendo os dados.

- **col_valores**: O nome da coluna que contém os valores a serem ponderados (preços no caso deste exemplo).
- **col_pesos**: O nome da coluna que contém os pesos associados a esses valores (pesos no caso deste exemplo).
- A função calcula a média ponderada multiplicando os valores pelos pesos, somando esses produtos e dividindo pelo somatório dos pesos. Ela retorna a média ponderada.

- **Criação do DataFrame produtos**:
- produtos = pd.DataFrame({'Nome_item': ['Chocolate', 'Chocolate', 'Chocolate', 'Biscoito', 'Biscoito', 'Biscoito', 'Sorvete', 'Sorvete', 'Sorvete'], 'Preço': [90, 50, 86, 87, 42, 48, 68, 92, 102], 'Peso': [4, 2, 3, 5, 6, 5, 3, 7, 5]})

- Este trecho cria um DataFrame chamado **produtos** que contém informações sobre diferentes produtos, incluindo o nome do item, o preço e o peso associado a cada item.

- **Cálculo da Média Ponderada Agrupada**:
- medias = produtos.groupby('Nome_item').apply(media_ponderada, 'Preço', 'Peso')

- Aqui, o código calcula a média ponderada agrupada pelo nome do item. Ele usa o método **groupby()** para agrupar os dados por **Nome_item** e, em seguida, aplica a função **media_ponderada** a cada grupo, passando as colunas **'Preço'** e **'Peso'** como argumentos. O resultado é armazenado no DataFrame **medias**.

- **Exibição das Médias Ponderadas**:
- for indice in range(len(medias)): print(f'Preço médio do {medias.index[indice]}: {medias[indice]:.2f}')

- Este laço de repetição percorre o DataFrame **medias** e exibe o preço médio de cada item, formatado com duas casas decimais.