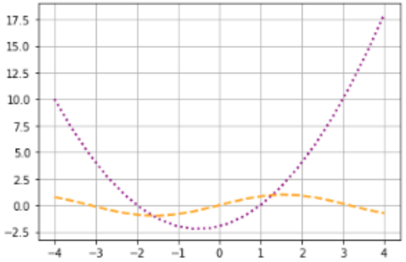
[+] EXERCÍCIO 1: [+]

[1]:

```python
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
x = np.linspace(-4,4,20)
y1 = np.sin(x)
y2 = x**2 + x - 2
# gera dois gráficos simultaneamente
# um com os valores (x,y1) e outro com (x,y2)
# armazenando o resultado em uma variável
line_1 = plt.plot(x,y1)
line_2=plt.plot(x,y2)
# o comando "setp" especifica o estilo para todos
plt.setp(line_1,color='orange',linewidth = 2, linestyle = '--')
plt.setp(line_2,color='purple',linewidth = 2, linestyle = ':')
plt.grid(True)
```



[+] TEOREMA DE BOLZANO:

[3]:

```python
import numpy as np
x = np.array([i for i in range(-4,5,1)]).reshape((1,9))
y=np.sin(x) - x**2 - x + 2;
print(x)
print(y)
```

```
[[-4 -3 -2 -1  0  1  2  3  4]]
[[ -9.2431975   -4.14112001  -0.90929743   1.15852902   2.
    0.84147098  -3.09070257  -9.85887999 -18.7568025 ]]
```

[+] RESOLVENDO AS RAÍZES LOCALIZADAS POR MEIO DO MÉTODO "fsolve":

[5]:

```python
from scipy.optimize import fsolve
from math import sin
def y(x):
    f=sin(x) - x**2 - x + 2
    return f
R1=fsolve(y,-1.5)
R2=fsolve(y,1.5)
print(f"Raiz 1 = {R1}")
print(f"Raiz 2 = {R2}")
```

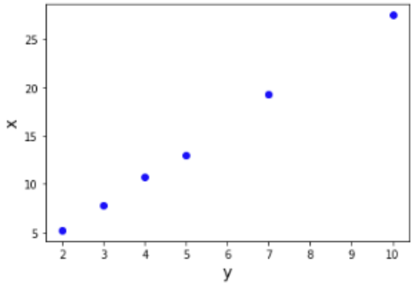
```
Raiz 1 = [-1.61854355]
Raiz 2 = [1.29203729]
```

```
[+] EXERCÍCIO 2: [+]
```

[7]:

```python
import numpy as np
import matplotlib.pyplot as plt

# define os dados
x = np.array([2,3,4,5,7,10])
y = np.array([5.2,7.8,10.7,13,19.3,27.5])
plt.plot(x, y, 'bo')
plt.ylabel("x", fontsize = 15)
plt.xlabel("y", fontsize = 15)
plt.show(True)
```



[+] COEFICIENTES DA REGRESSÃO LINEAR SIMPLES:

[17]:

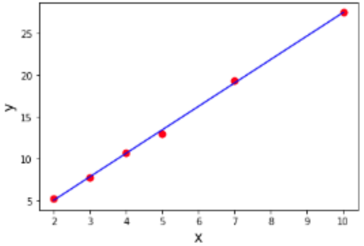
```python
n = np.size(x)
Sx= np.sum(x)
Sy=np.sum(y)
Sxy=np.sum(x*y)
Sxx=np.sum(x*x)
a1=(n*Sxy-Sx*Sy)/(n*Sxx-Sx**2)   #Calcula o coeficiente a1 da reta
a0=(Sxx*Sy-Sxy*Sx)/(n*Sxx-Sx**2) #Calcula o coeficiente a0 da reta
print(f"[+] Coeficiente Angular = {a1:.3f}")
print(f"[+] Coeficiente Linear = {a0:.3f}")
```

```
[+] Coeficiente Angular = 2.808
[+] Coeficiente Linear = -0.592
```

[21]:

```
# mostra os dados
plt.scatter(x, y, color = "r", marker = "o", s = 50)
  # prediz os valores
y_pred = a0 + a1*x
    # mostra a reta de regressão
plt.plot(x, y_pred, color = "b")

plt.xlabel('x', fontsize = 15)
plt.ylabel('y', fontsize = 15)
plt.show(True)
```



```
[+] RESÍDUO:
```

[24]:

```
#
def Residuo(x,y,b0,b1):
    n = len(y)
    RS = 0
    for i in range(0,n):
         y_pred=a0+a1*x[i]
         RS = RS + (y[i]-y_pred)**2
    return round(RS, 3)
print('[+] RS =', Residuo(x,y,a0,a1))
```

```
[+] RS = 0.292
```

```
[+] COEFICIENTE DE CORRELAÇÃO DE PEARSON:
```

[12]:

```
x = np.array([2,3,4,5,7,10])
y = np.array([5.2,7.8,10.7,13,19.3,27.5])
correlacao = np.corrcoef(x, y)
print(correlacao)
```

```
[[1.         0.99956783]
 [0.99956783 1.        ]]
```

```
[+] COEFICIENTE DE DETERMINAÇÃO R2:
```

[26]:

```
import sympy as sy
import numpy as np
from sklearn.linear_model import LinearRegression
x = np.array([2,3,4,5,7,10])
y = np.array([5.2,7.8,10.7,13,19.3,27.5])
x = x.reshape(-1, 1)
modelo = LinearRegression().fit(x, y)
print(modelo.intercept_)
print (modelo.coef_)
from sklearn.metrics import r2_score
R2 = r2_score(y, modelo.predict(x))
print('[+] R2 = ', R2)
print(f"%R2: {R2*100}")
```

```
-0.5922178988326898
[2.80817121]
[+] R2 =  0.9991358549470146
%R2: 99.91358549470146
```