Esercizio: Pacchetto `matplotlib`

Python fornisce il pacchetto matplotlib

...Per costruire grafici matematici

- Si tratta di un pacchetto molto esteso
- Noi ne useremo una piccola parte ...Ma se siete interessati potete leggere la documentazione online

I grafici vengono costruiti attraverso a chiamate a funzione

- Si usa una funzione per costruira un figura
- Funzioni per aggiungere elementi grafici
- Funzioni per modificare la presentazione del disegno

Vediamo come usarne alcune passo a passo

Preparazione di un Grafico

Innanzitutto va importato il sottopacchetto plt

In [1]:
```
from matplotlib import pyplot as plt
```

- Il sottopacchetto pyplot viene di solito rinominato come plt

Quindi, si costruisce una nuova figura con figure

In [2]:
```
plt.figure(figsize=(20, 3))
```

Out[2]:
```
<Figure size 2000x300 with 0 Axes>
```

```
<Figure size 2000x300 with 0 Axes>
```

- Si può specificare la dimensione con l'argomento figsize
- La dimensione è una tupla nel formato (larghezza, altezza)

Funzione `plot`

La funzione di disegno plot permette di disegnare curve

Proveremo ad usarla per disegnare la funzione: $f(x) = \sin(x) + 0.1x$

- Innanzitutto, costruiamo due array con le coordinate x e y dei punti sulla curva

In [3]:
```
import numpy as np

x = np.linspace(0, 10, 12)
y = np.sin(x) + 0.1 * x
n = 4
print(f'Primi {n} elementi di x: {x[:n]}')
print(f'Primi {n} elementi di y: {y[:n]}')
```

```
Primi 4 elementi di x: [0.         0.90909091 1.81818182 2.72727273]
Primi 4 elementi di y: [0.         0.87985455 1.15137413 0.67529476]
```

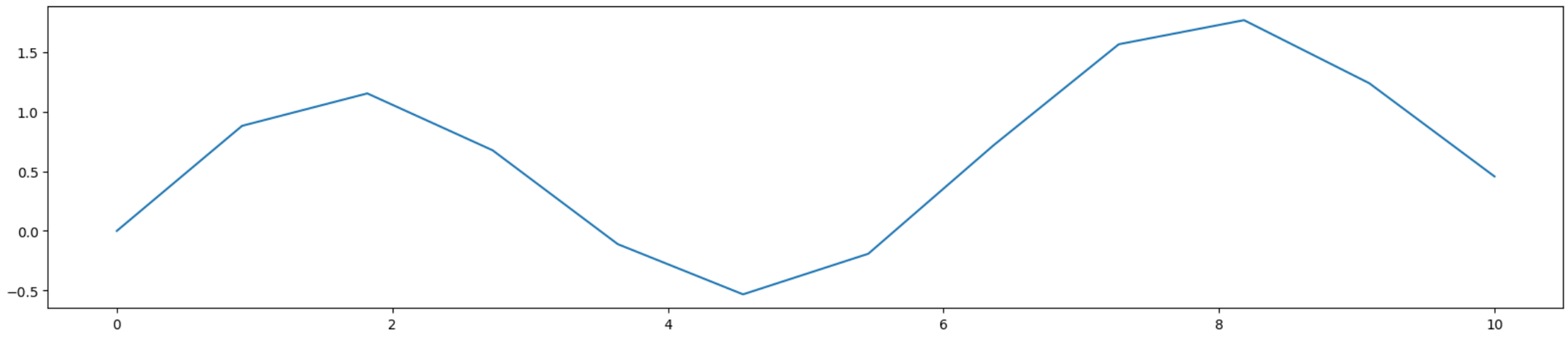
- Nel codice utilizziam la funzione `linspace` per definire una serie di valori
- ...Quindi usiamo le operazioni di `numpy` per valutare la funzione

Funzione `plot`

Ora possiamo invocare plot(x, y, ...) per disegnare la curva

- Gli argomenti `x` e `y` sono due collezioni con le coordinate di punti sulla curva
- La funzione ottiene un disegno collegando con una retta i punti adiacenti

In [4]:
```
plt.figure(figsize=(20, 4))
plt.plot(x, y)
plt.show()
```

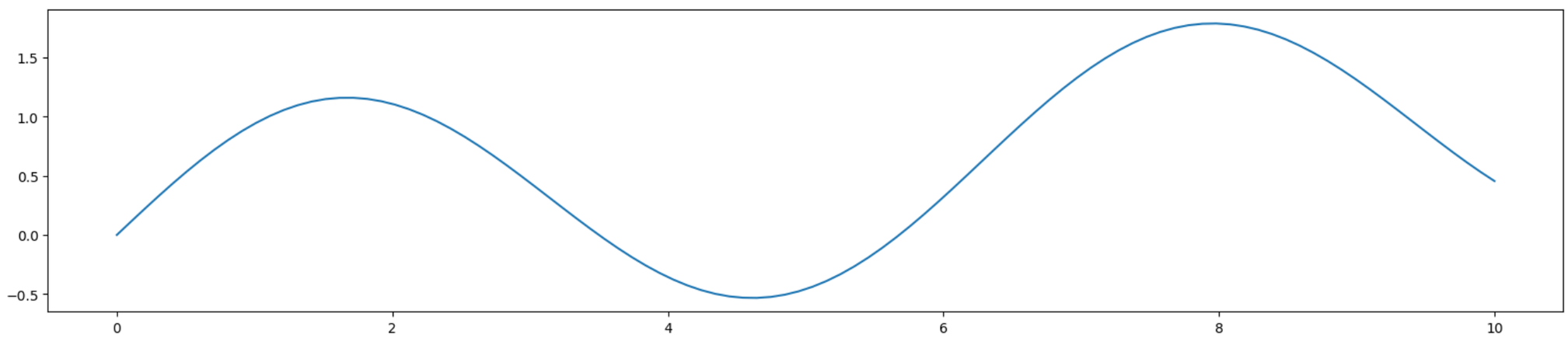


Funzione `plot`

Aumentando il numero di punti per unità possiamo migliorare il disegno

In [5]:
```
x = np.linspace(0, 10, 100)
y = np.sin(x) + 0.1 * x

plt.figure(figsize=(20, 4))
plt.plot(x, y)
plt.show()
```

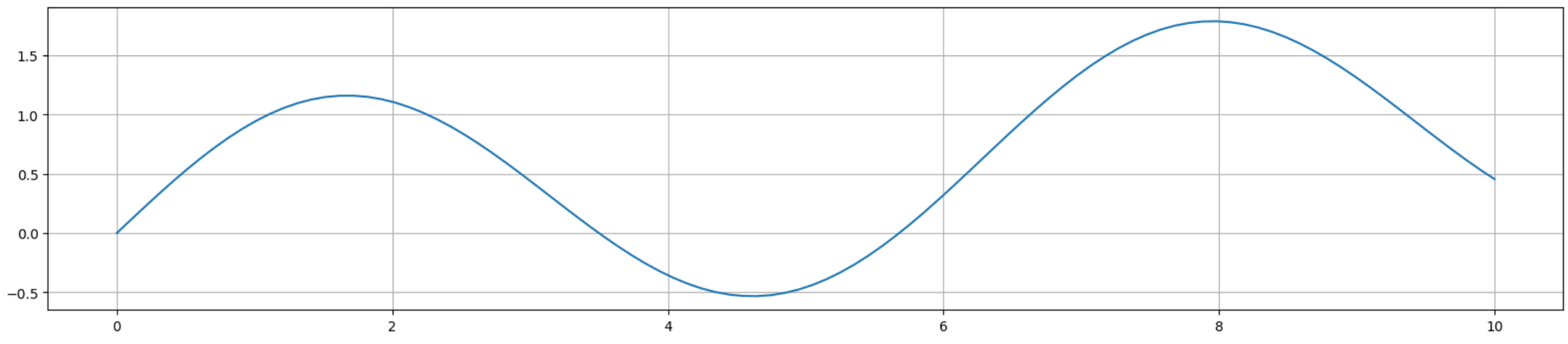


Funzione `plot`

Possiamo aggiungere una griglia con `grid`

In [6]:
```
x = np.linspace(0, 10, 100)
y = np.sin(x) + 0.1 * x

plt.figure(figsize=(20, 4))
plt.plot(x, y)
plt.grid()
plt.show()
```

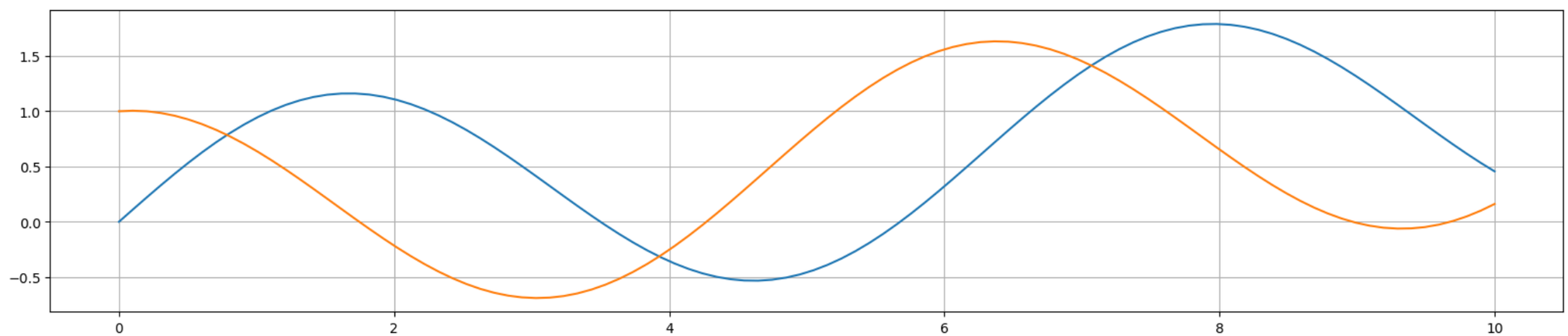


Funzione `plot`

Si può disegnare più di una curva sullo stesso grafico

In [7]:
```
x = np.linspace(0, 10, 100)
y = np.sin(x) + 0.1 * x
y2 = np.cos(x) + 0.1 * x

plt.figure(figsize=(20, 4))
plt.plot(x, y)
plt.plot(x, y2) # Disegno la seconda curva
plt.grid()
plt.show()
```

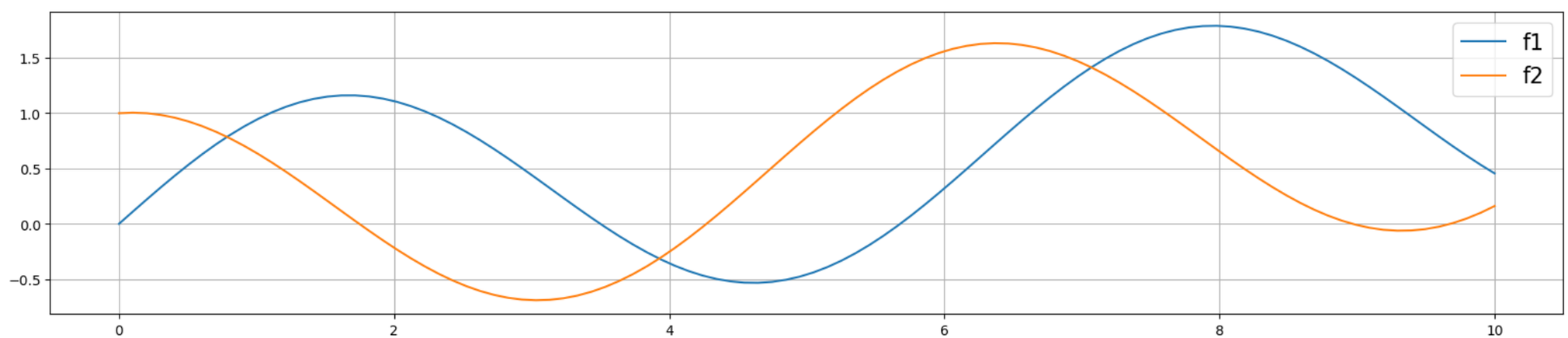


Funzione `plot`

Si può assegnare un nome (label) ad ogni curve, quindi stampare una legenda

In [8]:
```
x = np.linspace(0, 10, 100)
y = np.sin(x) + 0.1 * x
y2 = np.cos(x) + 0.1 * x

plt.figure(figsize=(20, 4))
plt.plot(x, y, label='f1')
plt.plot(x, y2, label='f2')
plt.grid()
plt.legend(fontsize=16) # Aggiungo una legenda
plt.show()
```



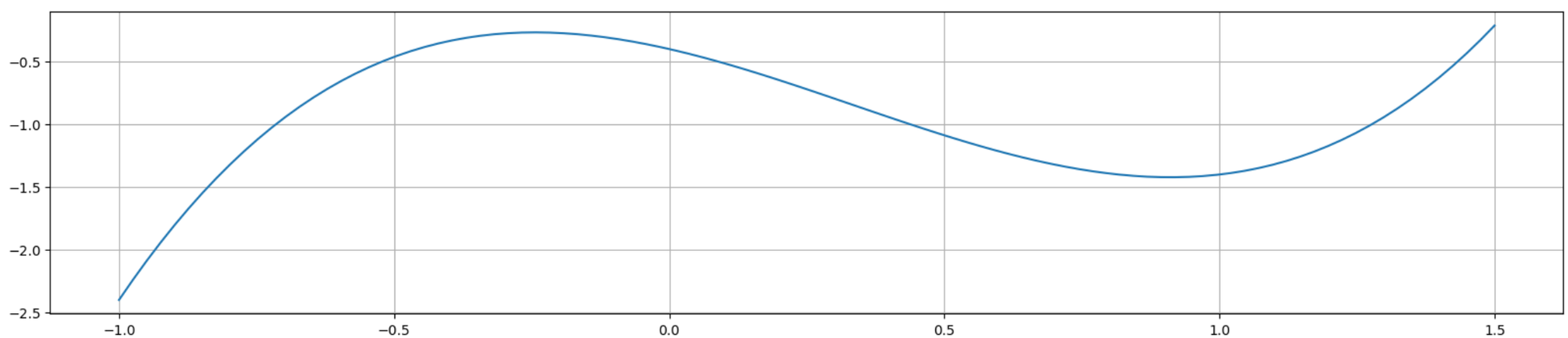
Esercizio

Provare a graficare la funzione $1.5x^3 - 1.5x^2 - x - 0.4$

- Disegnatela nell'intervallo $[-1, 1.5]$, con diverse densità di punti

In [12]:
```
x = np.linspace(-1, 1.5, 100)
y = 1.5*x**3 - 1.5*x**2 - x - 0.4

plt.figure(figsize=(20, 4))
plt.plot(x, y)
plt.grid()
plt.show()
```



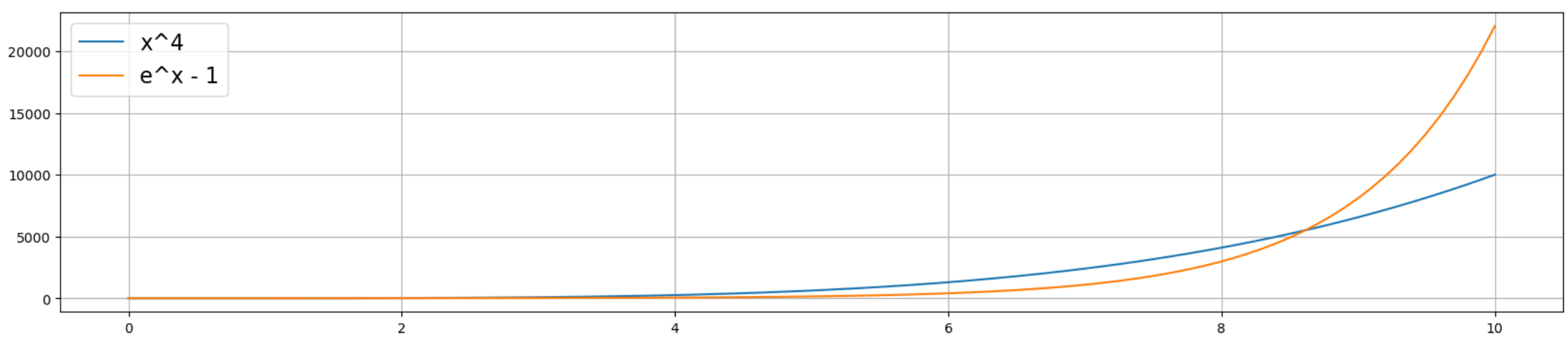

Esercizio

Confrontate sull'intervallo $[0, 10]$

...Le funzioni x^4 ed $e^x - 1$

In [20]:
```
x = np.linspace(0, 10, 100)
y1 = x**4
y2 = np.exp(x) - 1

plt.figure(figsize=(20, 4))
plt.plot(x, y1, label='x^4')
plt.plot(x, y2, label='e^x - 1')
plt.grid()
plt.legend(fontsize=16)
plt.show()
```



Esercizio

Determinate visivamente per quali valori di x seguente equazione è soddifatta

$$\sin(x) - \frac{1}{2}(e^x - 1)$$

- Procedete disegnando la funzione ed osservate in quali punti si azzera

In [27]:
```
x = np.linspace(-0.5, 1.25, 100)
y = np.sin(x) - 0.5*(np.exp(x) - 1)

plt.figure(figsize=(20, 4))
plt.plot(x, y)
plt.grid()
plt.show()
```