In [25]:
```
import ipywidgets as widgets
import numpy as np
import matplotlib.pyplot as plt
```

Esercizio 1

In [26]:
```
def power(n):
    x = np.array([0,1,2,3,4,5,6])
    y = x**n + 5
    return plt.plot(x,y,'-o')
    # plt.show()
widgets.interactive(power, n=(0,7))
```

Out[26]:
```
interactive(children=(IntSlider(value=3, description='n', max=7), Output()), _dom_
classes=('widget-interact',)…
```

In [27]:
```
def power_wrong(n):
    x = [0,1,2,3,4,5,6]
    y = x**n + 5
    return plt.plot(x,y,'-o')
    # plt.show()
widgets.interactive(power, n=(0,7))
```

Out[27]:
```
interactive(children=(IntSlider(value=3, description='n', max=7), Output()), _dom_
classes=('widget-interact',)…
```

In [28]:
```
type(x)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[28], line 1
----> 1 type(x)

NameError: name 'x' is not defined
```

In [29]:
```
w = np.array([0,1,2,3,4,5,6])
z = [0,1,2,3,4,5,6]
u = 3
type(w), type(z)
```

Out[29]:
```
(numpy.ndarray, list)
```

In [30]:
```
print(w,z)
```

```
[0 1 2 3 4 5 6] [0, 1, 2, 3, 4, 5, 6]
```

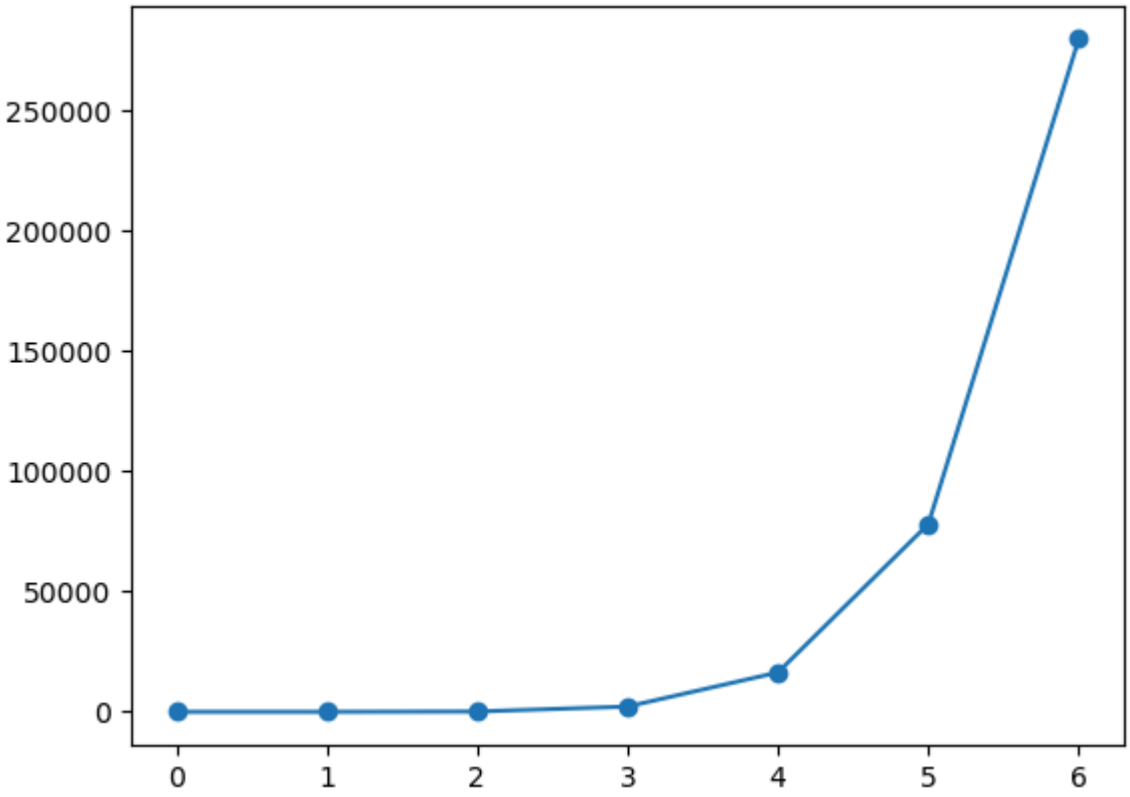
In [31]:
```
def power_bis(n):
    x = [0,1,2,3,4,5,6]
    y = np.power(x,n)+5
    return plt.plot(x,y,'-o')
```

In [32]:
```
widgets.interactive(power_bis, n=(0,7))
```

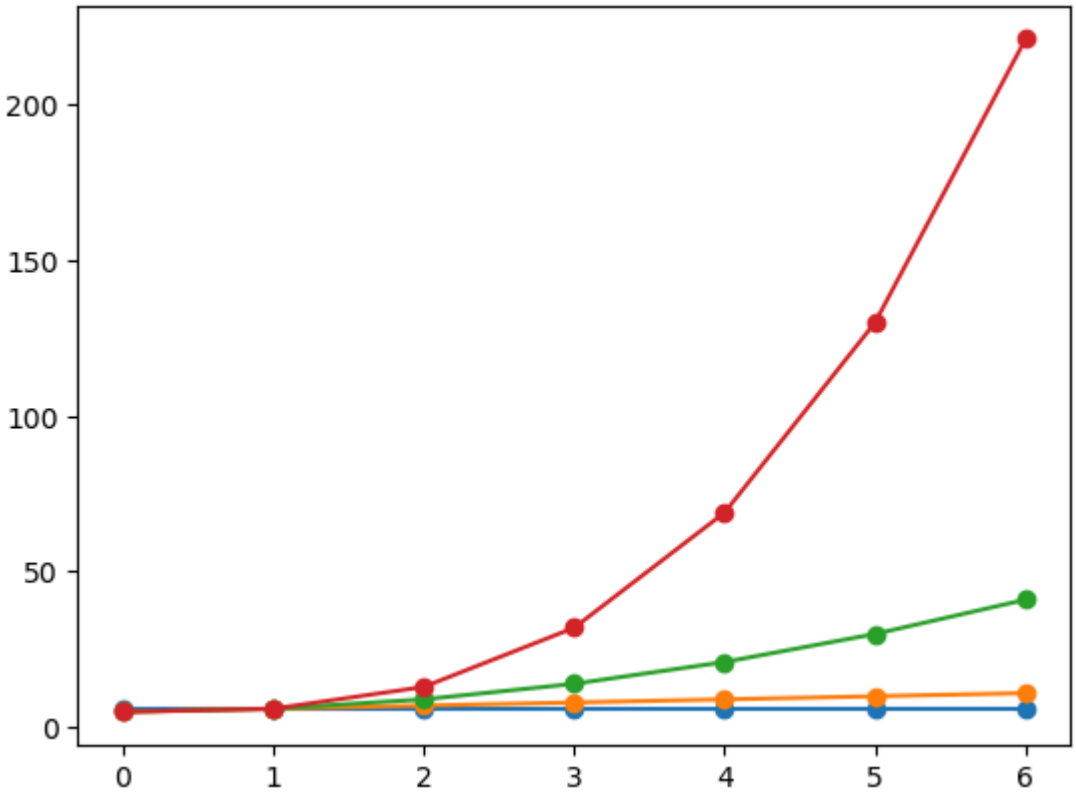
Out[32]:
```
interactive(children=(IntSlider(value=3, description='n', max=7), Output()), _dom_
classes=('widget-interact',)…
```

In [33]:
```
power_bis(7)
```

Out[33]:
```
[<matplotlib.lines.Line2D at 0x1b7b6458610>]
```



In [34]:
```
for n in range(4):
    power_bis(n)
```



Qualche informazione sulla funzione intrinseca `print`.

In [35]:
```
print(w,z,end=' ')
print(u)
```

```
[0 1 2 3 4 5 6] [0, 1, 2, 3, 4, 5, 6] 3
```

Esercizio 2 (Programmare in Python: controllo del flusso)

- Scrivere un programma che stampi tutti i divisori di un numero *n* inizializzato nel programma stesso.
- Modificare il programma perché stampi solo i divisori di *n* che sono numeri primi.

Codice per trovare i divisori di un dato numero *n*

In [36]:
```
n = 1000
for i in range(2, n//2+1):
    if n%i == 0: print(i)
```

```
2
4
5
8
10
20
25
40
50
100
125
200
250
500
```

In [37]:
```
n = 1000
dv = []
for i in range(2, n//2+1):
    if n%i == 0:
        dv.append(i)
print(dv)
```

```
[2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 125, 200, 250, 500]
```

Codice per trovare i numeri primi divisori di *n*

In [38]:
```
n = 1000
pdv = []
for i in range(2, n//2+1):
    if n%i == 0:
        for j in range(2, i//2+1):
            if i%j == 0: break
        else:
            pdv.append(i)
print(pdv)
```

```
[2, 5]
```

Esercizio 3 (Python: Funzioni e Moduli)

- Scrivere una funzione che produce tutti i divisori di un numero dato in ingresso, a partire dal codice creato precedentemente.
- Scrivere una funzione che valuta se un numero (dato come input) è primo o meno.

Funzione per trovare i divisori di un numero *n*

In [39]:
```
def divisors(n):
    dv = []
    for i in range(2, n//2+1):
        if n%i == 0:
```

```
            dv.append(i)
    return dv
```

In [40]:
```
divisors(300)
```

Out[40]:
```
[2, 3, 4, 5, 6, 10, 12, 15, 20, 25, 30, 50, 60, 75, 100, 150]
```

Funzione che verifica se *n* è primo

Modo 1: da scratch

In [41]:
```
def is_prime(n):
    for j in range(2, n//2+1):
        if n%j == 0: return False
    return True
```

In [42]:
```
is_prime(30), is_prime(31)
```

Out[42]:
```
(False, True)
```

Modo 2: sfruttando la funzione divisors()

In [43]:
```
def is_prime(n):
    if divisors(n): return False
    return True
```

In [44]:
```
is_prime(45), is_prime(37)
```

Out[44]:
```
(False, True)
```

Esercizio 4 (Python: Funzioni e Moduli)

- Creare un modulo `divisor_is_prime.py` contenente le due funzioni costruite nell'esercizio precedente e provare le istruzioni riportate sopra.
- Creare un programma che, dopo aver definito in qualche modo (un modo a scelta) una lista di numeri, la scorra sfruttando il modulo `divisor_is_prime.py` per verificare la natura di numero primo di ciascun elemento della lista e fornire in output la lista del sottoinsieme di numeri primi.

Esercizio 5 (Python: Classi)

Creare una classe `persona` con gli attributi `nome` ed `età` e dotarla di metodi che ne stampino gli attributi.

In [ ]: