Let's create one dataset which shows the number of Cars which was sold

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
cars={'Audi':199, 'BMW':89, 'Mercedes Benz':67, 'Harrier':101}
cars
```

Out[4]:
```
{'Audi': 199, 'BMW': 89, 'Mercedes Benz': 67, 'Harrier': 101}
```

In [7]:
```
type(cars)
```

Out[7]:
```
dict
```

In [17]:
```
labels=list(cars.keys())
values=list(cars.values())
```

In [18]:
```
labels
```

Out[18]:
```
['Audi', 'BMW', 'Mercedes Benz', 'Harrier']
```

In [19]:
```
values
```

Out[19]:
```
[199, 89, 67, 101]
```

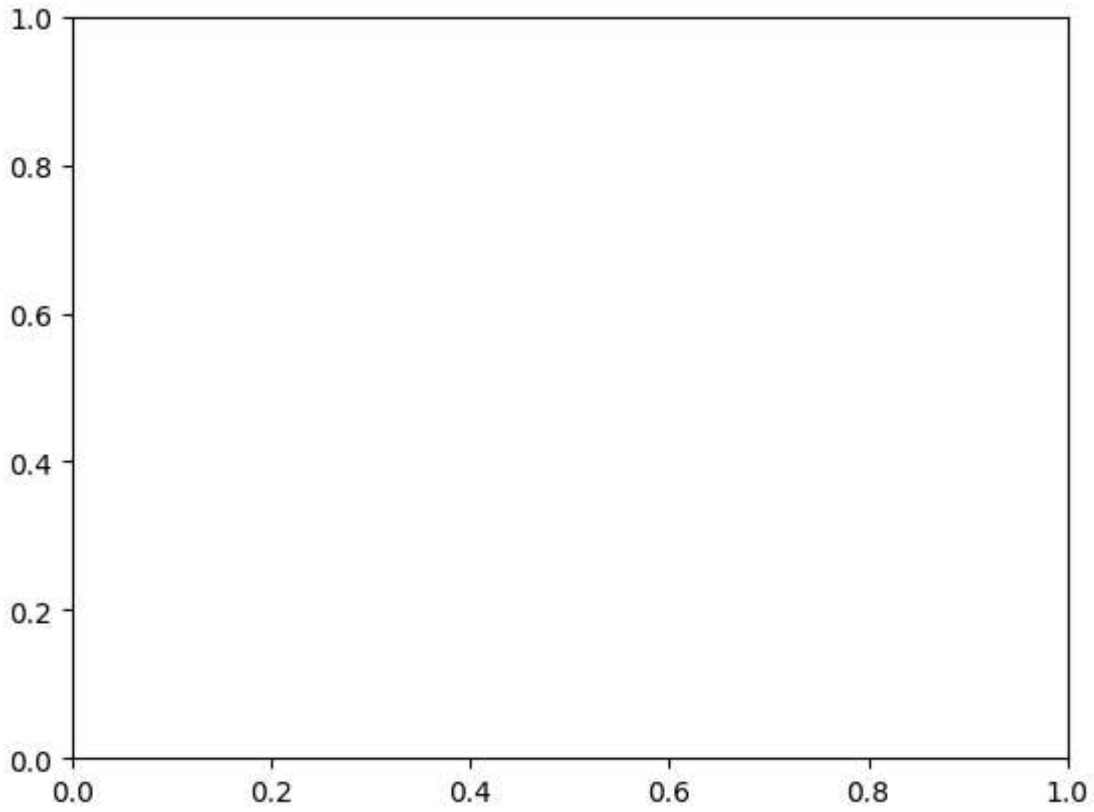
In [20]:
```
plt.bar(range(labels, values))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 1
----> 1 plt.bar(range(labels, values))

TypeError: 'list' object cannot be interpreted as an integer
```

In [21]:
```
plt.bar(range(len(cars)), cars)
```

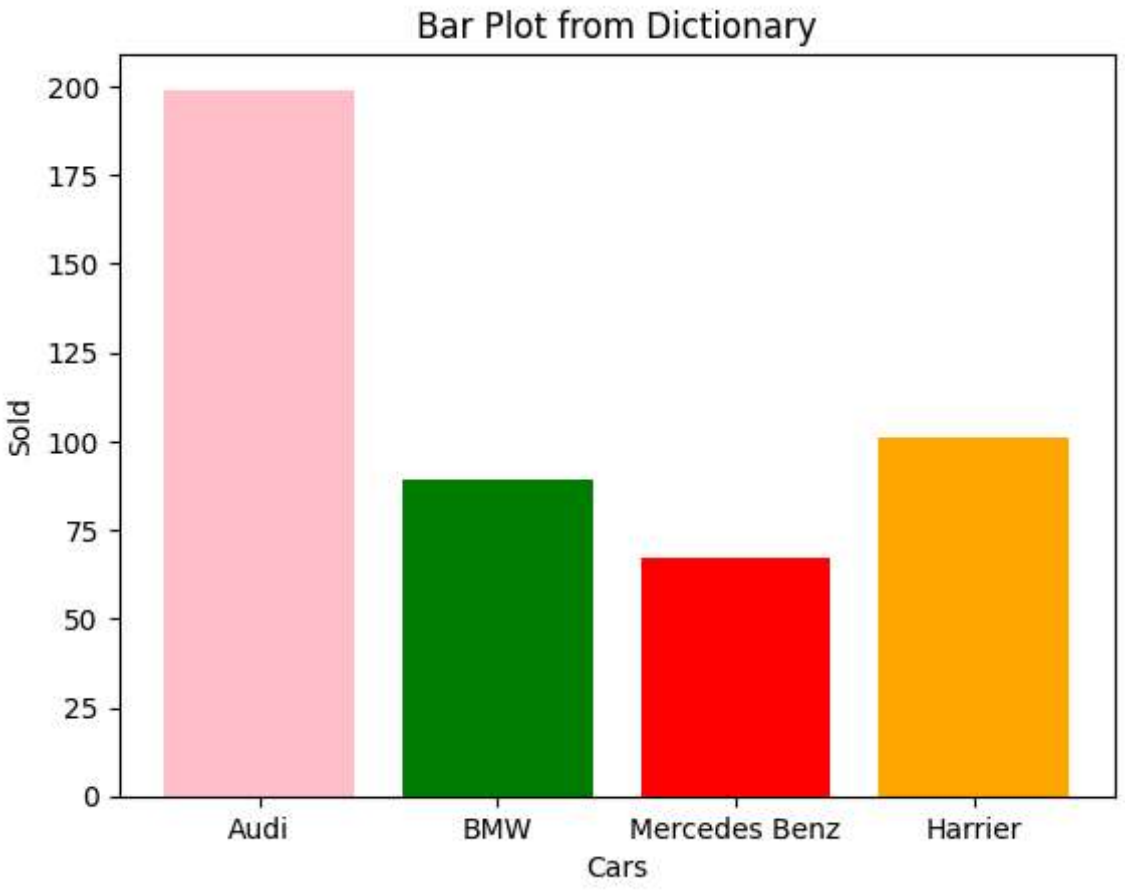
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[21], line 1
----> 1 plt.bar(range(len(cars)), cars)

File ~\anaconda3\Lib\site-packages\matplotlib\pyplot.py:2439, in bar(x, height, widt
h, bottom, align, data, **kwargs)
   2435 @_copy_docstring_and_deprecators(Axes.bar)
   2436 def bar(
   2437         x, height, width=0.8, bottom=None, *, align='center',
   2438         data=None, **kwargs):
-> 2439     return gca().bar(
   2440         x, height, width=width, bottom=bottom, align=align,
   2441         **({"data": data} if data is not None else {}), **kwargs)

File ~\anaconda3\Lib\site-packages\matplotlib\__init__.py:1459, in _preprocess_data.
<locals>.inner(ax, data, *args, **kwargs)
   1456 @functools.wraps(func)
   1457 def inner(ax, *args, data=None, **kwargs):
   1458     if data is None:
-> 1459         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1461     bound = new_sig.bind(ax, *args, **kwargs)
   1462     auto_label = (bound.arguments.get(label_namer)
   1463                   or bound.kwargs.get(label_namer))

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:2384, in Axes.bar(self,
x, height, width, bottom, align, **kwargs)
   2381         x = 0
   2383 if orientation == 'vertical':
-> 2384     self._process_unit_info(
   2385         [("x", x), ("y", height)], kwargs, convert=False)
   2386     if log:
   2387         self.set_yscale('log', nonpositive='clip')

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:2549, in _AxesBase._proc
ess_unit_info(self, datasets, kwargs, convert)
   2547     # Update from data if axis is already set but no unit is set yet.
   2548     if axis is not None and data is not None and not axis.have_units():
-> 2549         axis.update_units(data)
   2550 for axis_name, axis in axis_map.items():
   2551     # Return if no axis is set.
   2552     if axis is None:

File ~\anaconda3\Lib\site-packages\matplotlib\axis.py:1713, in Axis.update_units(sel
f, data)
   1711 neednew = self.converter != converter
   1712 self.converter = converter
-> 1713 default = self.converter.default_units(data, self)
   1714 if default is not None and self.units is None:
   1715     self.set_units(default)

File ~\anaconda3\Lib\site-packages\matplotlib\category.py:105, in StrCategoryConvert
er.default_units(data, axis)
    103 # the conversion call stack is default_units -> axis_info -> convert
    104 if axis.units is None:
--> 105     axis.set_units(UnitData(data))
```

```
    106 else:
    107     axis.units.update(data)

File ~\anaconda3\Lib\site-packages\matplotlib\category.py:181, in UnitData.__init__
(self, data)
    179 self._counter = itertools.count()
    180 if data is not None:
--> 181     self.update(data)

File ~\anaconda3\Lib\site-packages\matplotlib\category.py:214, in UnitData.update(se
lf, data)
    212 # check if convertible to number:
    213 convertible = True
--> 214 for val in OrderedDict.fromkeys(data):
    215     # OrderedDict just iterates over unique values in data.
    216     _api.check_isinstance((str, bytes), value=val)
    217     if convertible:
    218         # this will only be called so long as convertible is True.

TypeError: unhashable type: 'dict'
```

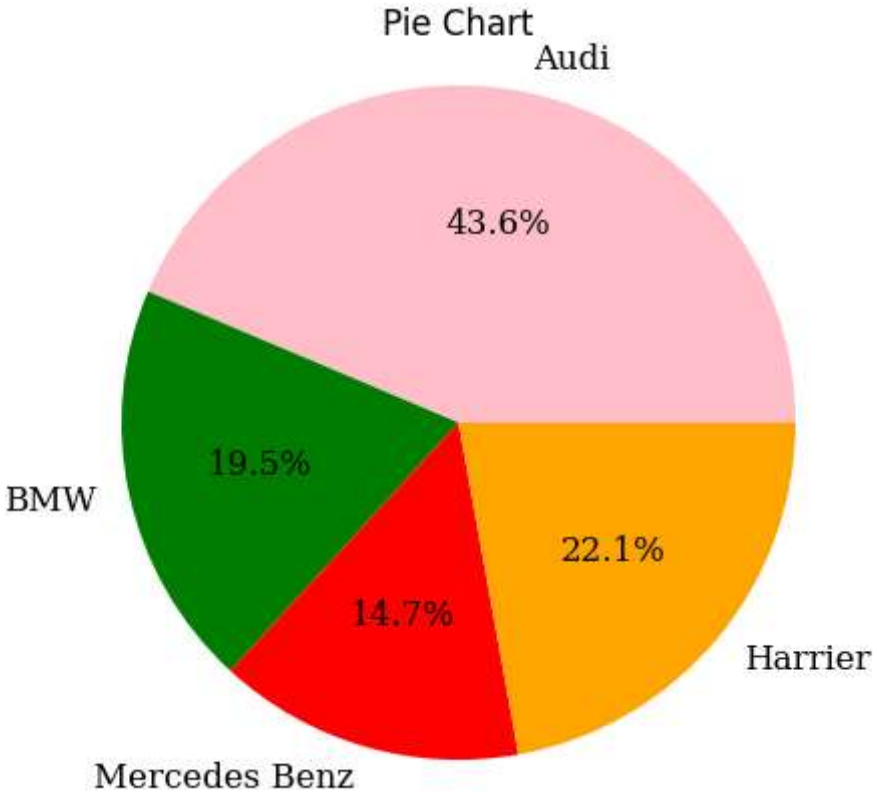


In [26]:
```
# Create numeric x values
x = range(len(labels))

colors = ['pink', 'green', 'red', 'orange']

# Create bar plot
plt.bar(x, values, tick_label=labels, color=colors)

plt.xlabel('Cars')
plt.ylabel('Sold')
```

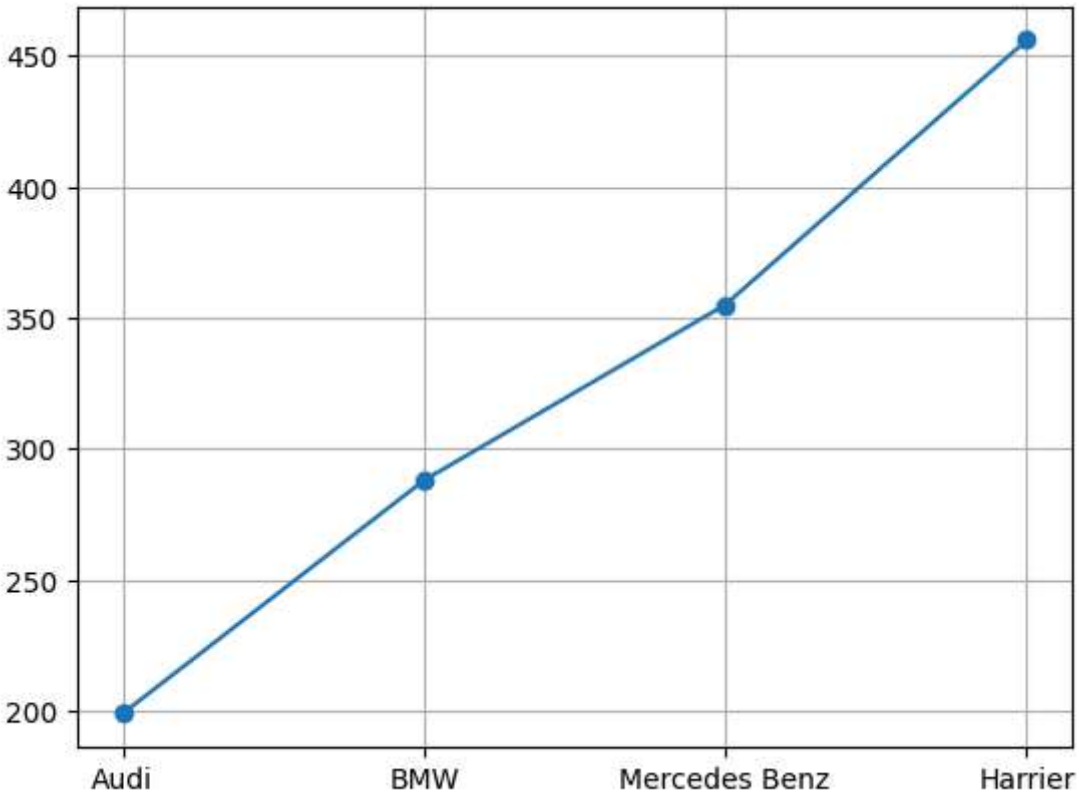
```
plt.title('Bar Plot from Dictionary')
plt.show()
```



In [43]:
```
# Define font style
font_style = {'fontsize': 12, 'fontweight': 'normal', 'fontfamily': 'serif'}

# Create pie chart
plt.pie(values, labels=labels, autopct='%1.1f%%', colors=colors, textprops=font_sty

#customize colors
colors = ['pink', 'green', 'red', 'orange']

# Add title
plt.title('Pie Chart')

# Show plot
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```

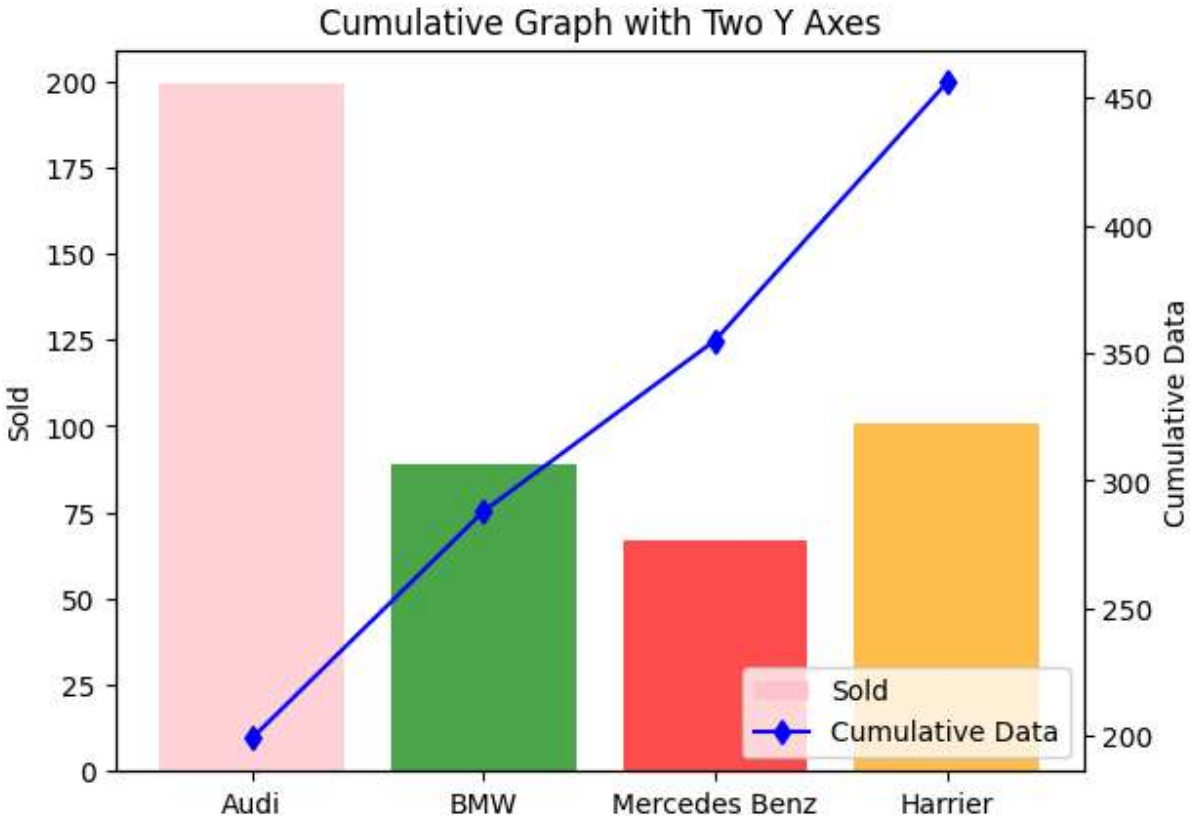


In [45]:
```
# Calculate cumulative sum
cumulative_values = [sum(values[:i+1]) for i in range(len(values))]

# Create cumulative graph
plt.plot(labels, cumulative_values, marker='o', linestyle='-')

plt.grid(True)
plt.show()
```



In [58]:
```
import numpy as np

# Extract keys and values
labels = list(cars.keys())
values = np.array(list(cars.values()))


# Calculate cumulative sum
cumulative_values = np.cumsum(values)

# Create figure and axes
fig, ax1 = plt.subplots()

# Plot original data on the first y-axis
ax1.bar(labels, values, color=('pink','green','red','orange'), alpha=0.7, label='So
ax1.set_ylabel('Sold')

# Create a secondary y-axis
ax2 = ax1.twinx()

# Plot cumulative data on the secondary y-axis
ax2.plot(labels, cumulative_values, color='b', marker='d', label='Cumulative Data')
ax2.set_ylabel('Cumulative Data')

# Add legend
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc='lower right')

# Add title
plt.title('Cumulative Graph with Two Y Axes')
```

```
# Show plot
plt.show()
```



In [ ]: