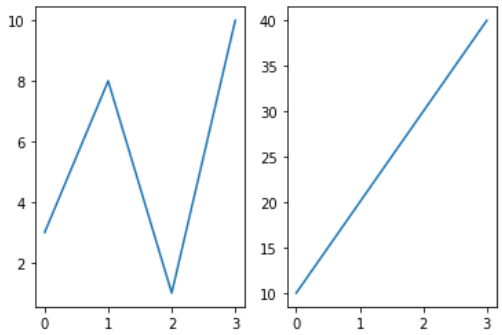
Assignment 23 Solutions

Q1. If you have any, what are your choices for increasing the comparison between different figures on the same graph?

Ans: Matplotlib provides a convenient method called subplots for increasing the comparison between different figures on the same graph. Subplots mean a group of smaller axes (where each axis is a plot) that can exist together within a single figure.

In [1]:

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

#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```



Q2. Can you explain the benefit of compound interest over a higher rate of interest that does not compound after reading this chapter?

Ans: Compound interest makes a sum of money grow at a faster rate than simple interest,because in addition to earning returns on the money you invest, you also earn returns on those returns at the end of every compounding period, which could be daily, monthly, quarterly or annually.

Q3. What is a histogram, exactly? Name a numpy method for creating such a graph.

Ans: A histogram displays numerical data by grouping data into "bins" of equal width. Each bin is plotted as a bar whose height corresponds to how many data points are in that bin. Bins are also sometimes called "intervals", "classes", or "buckets".

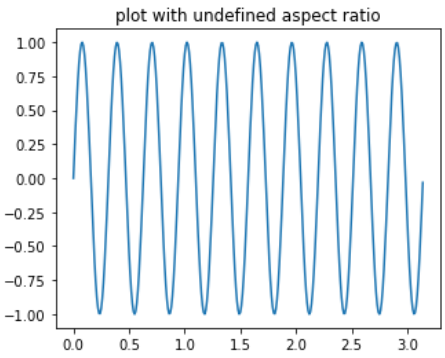
`NumPy.histogram()` is the built in function used.

Q4. If necessary, how do you change the aspect ratios between the X and Y axes ?

Ans: We can use `figure(figsize=(10,8))` function inside the `matplot.pyplot` library which we scale down or up the graph.

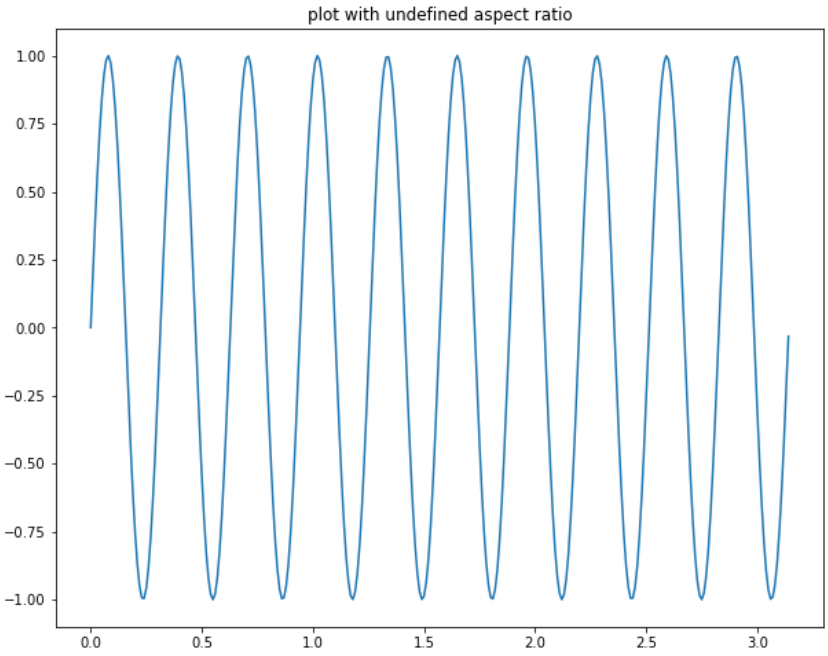
In [3]:

```
import matplotlib.pyplot as plt
import numpy as np
import math
x = np.arange(0,math.pi,0.01)
y = np.sin(20*x)

plt.figure(figsize = (5,4))
plt.plot(x,y)
plt.title('plot with undefined aspect ratio')
plt.show()
```



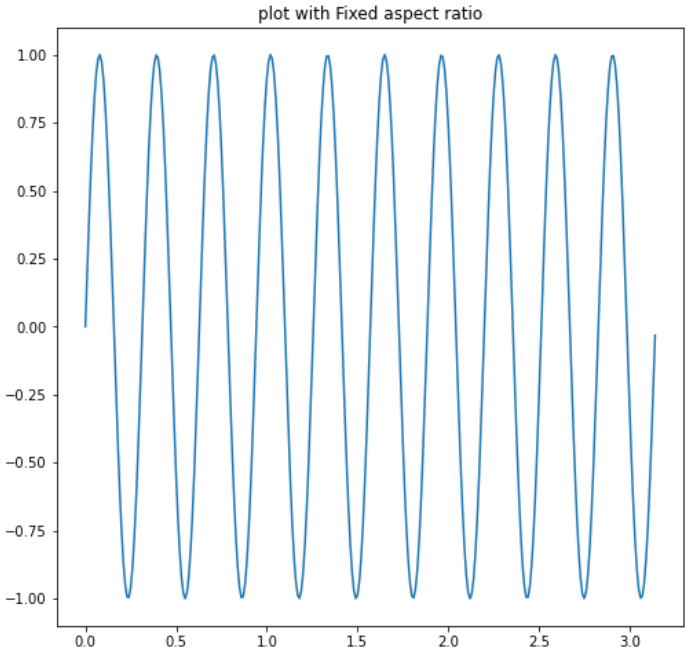
In [4]:

```
plt.figure(figsize=(10,8))
plt.plot(x,y)
plt.title('plot with undefined aspect ratio')
plt.show()
```



In [5]:

```
plt.figure(figsize=(10,8)) #fig size same as before
ax = plt.gca() #we first need to get the axis handle
ax.set_aspect(1.5) #sets the height to width ratio to 1.5.
#Use help(ax.set_aspect) for more options.

plt.plot(x,y)
plt.title('plot with Fixed aspect ratio')
plt.show()
```



Q5. Compare and contrast the three types of array multiplication between two numpy arrays: dot product, outer product, and regular multiplication of two numpy arrays ?

Ans: :In Python if we have two numpy arrays which are often referred as a vector. The '*' operator and numpy.dot() work differently on them.

- In regular multiplication values of same index get multiplied.
- In dot product there is row wise multiplication, row of one array with column of second array and so on.
- In outer multiplication every element of first array a1 will be multiply by every element of other array a2 such such the number of columns will be equal to the number of element in another array a2.

In [6]:

```
import numpy as np


# vector v1 of dimension (2, 2)
v1 = np.array([[1, 2], [1, 2]])

# vector v2 of dimension (2, 2)
v2 = np.array([[1, 2], [1, 2]])

print("vector multiplication")
print(np.dot(v1, v2))
print("outer multiplication")
print(np.outer(v1, v2))
print("\nElementwise multiplication of two vector")
print(v1 * v2)

```

```
vector multiplication
[[3 6]
 [3 6]]
outer multiplication
[[1 2 1 2]
 [2 4 2 4]
 [1 2 1 2]
 [2 4 2 4]]

Elementwise multiplication of two vector
[[1 4]
 [1 4]]
```

In [7]:

```
import numpy as np


v1 = np.array([[1, 2, 3], [1, 2, 3], [1, 2, 3]])

v2 = np.array([[[1, 2, 3], [1, 2, 3], [1, 2, 3]]])

print("vector multiplication")
print(np.dot(v1, v2))
print("outer multiplication")
print(np.outer(v1, v2))
print("\nElementwise multiplication of two vector")
print(v1 * v2)

```

```
vector multiplication
[[[ 6 12 18]]

 [[ 6 12 18]]

 [[ 6 12 18]]]
outer multiplication
[[1 2 3 1 2 3 1 2 3]
 [2 4 6 2 4 6 2 4 6]
 [3 6 9 3 6 9 3 6 9]
 [1 2 3 1 2 3 1 2 3]
 [2 4 6 2 4 6 2 4 6]
 [3 6 9 3 6 9 3 6 9]
 [1 2 3 1 2 3 1 2 3]
 [2 4 6 2 4 6 2 4 6]
 [3 6 9 3 6 9 3 6 9]]

Elementwise multiplication of two vector
[[[1 4 9]
  [1 4 9]
  [1 4 9]]]
```

Q6. Before you buy a home, which numpy function will you use to measure your monthly mortgage payment ?

Ans: `np.pmt(rate, nper, pv)` function we will be using in order to calculate monthly mortgage payment before you purchase a house.

- `rate` = The periodic interest rate
- `nper` = The number of payment periods
- `pv` = The total value of the mortgage loan

Q7. Can string data be stored in numpy arrays? If so, list at least one restriction that applies to this data ?

Ans: Yes, an array can store the string. The limitation which imposed on the string data is, whenever we store the data of string dtype then we should keep in mind that the string which is having the maximum length will become the limit. Below example demonstrates that when New Zealand is added then it takes only 'New Z' because limit is set as 5 before.

In [8]:

```
# importing numpy as np
import numpy as np

# Create the numpy array
country = np.array(['USA', 'Japan', 'UK', '', 'India', 'China'])

# Print the array
print(country)

# Assign 'New Zealand' at the place of missing value
country[country == ''] = 'New Zealand'

# Print the modified array
print(country)
```

```
['USA' 'Japan' 'UK' '' 'India' 'China']
['USA' 'Japan' 'UK' 'New Z' 'India' 'China']
```