Applied: Exercise 15

Import notebook functions

```
from notebookfuncs import *
```

Exercise 15

This problem involves writing functions.

(a)

Write a function, `Power()`, that prints out the result of raising 2 to the 3_{rd} power. In other words, your function should compute 2^3 and print out the results.

Hint: Recall that x^a raises x to the power a. Use the print() function to display the result.

```
def Power():
  print(2 ** 3)

Power()
```

```
8
```

(b)

Create a new function, `Power2()`, that allows you to pass any two numbers, x and a, and prints out the value of x^a. You can do this by beginning your function with the line

```
def Power2(x, a):
```

You should be able to call your function by entering, for instance, `Power2(3, 8)` on the command line. This should output the value of 3^8 , namely, 6,561.

```
def Power2(x, a):
  print(x ** a)

Power2(3, 8)
```

```
6561
```

(c)

Using the `Power2()` function that you just wrote, compute 10^3 , 8^{17} , and 131^3 .

```
Power2(10,3)
Power2(8,17)
Power2(131, 3)
```

```
1000
2251799813685248
2248091
```

(d)

Now create a new function, Power3(), that actually returns the result x^a as a Python object, rather than simply printing it to the screen. That is, if you store the value x^a in an object called result within your function, then you can simply return this result, using the following line:

```
return result
```

Note that the line above should be the last line in your function, and it should be indented 2 or 4 spaces, based on your preference.

```
def Power3(x, a):
  result = x ** a
  return result
```

(e)

Now using the `Power3()` function, create a plot of $f(x) = x^2$. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x^2 . Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the log-scale. You can do this by using the `ax.set_xscale()` and `ax.set_yscale()` methods of the axes you are plotting to.

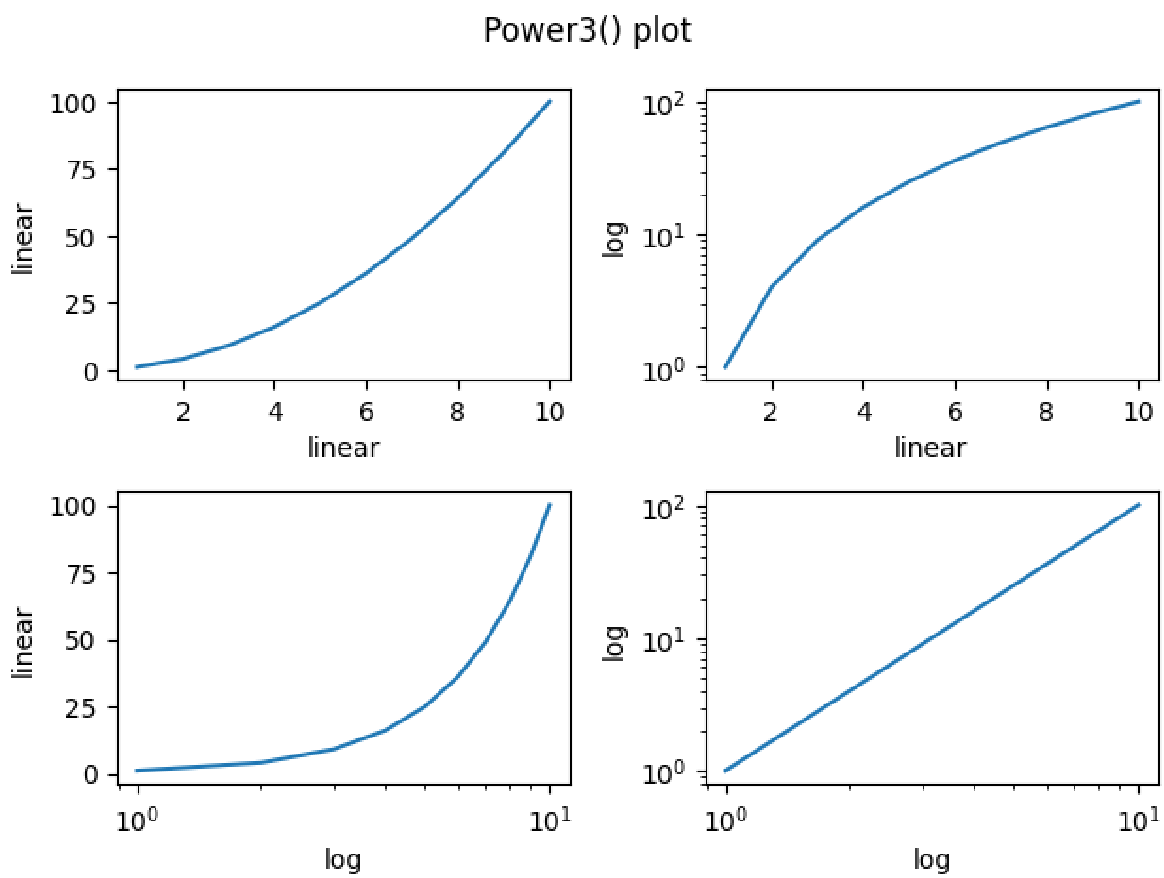
```
import matplotlib.pyplot as plt
import numpy as np
scales = ["linear", "linear-log",'log-linear', 'log-log']
fig, axs = plt.subplot_mosaic([scales[0:2],
                               scales[2:]], layout='tight')

fig.suptitle("Power3() plot")
fig.subplots_adjust(hspace=0.5)
fig.subplots_adjust(wspace=0.5)
x = np.arange(1, 11, 1)
y = Power3(x, 2)

def get_labels(scales):
  if scales is None:
    return None
  labels = []
  for scale in scales:
    arr = scale.split('-')
    if len(arr) == 1:
      labels.append(tuple((arr[0], arr[0])))
    else:
      labels.append(tuple((arr[0], arr[1])))
  return labels

labels = get_labels(scales)
```

```
for ax, scale, label in zip(axs.values(), scales, labels):
  ax = axs[scale]
  ax.plot(x, y)
  ax.set_xlabel(label[0])
  ax.set_ylabel(label[1])
  ax.set_xscale(label[0])
  ax.set_yscale(label[1])
```



(f)

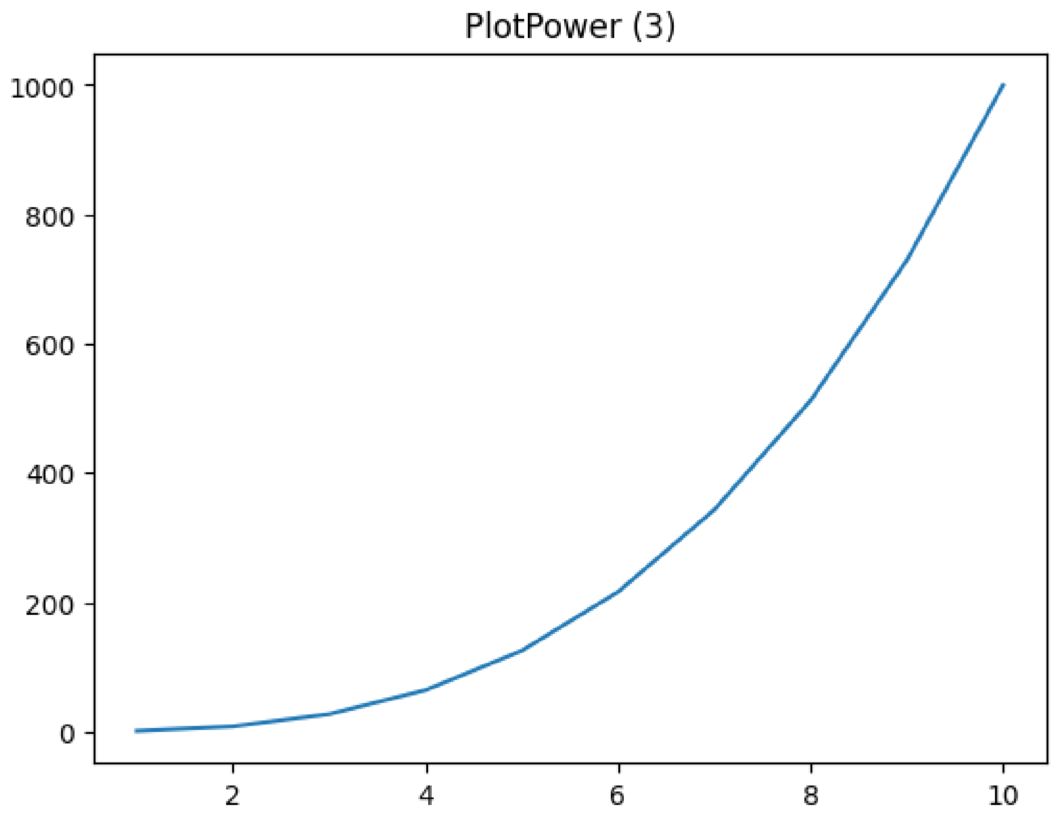
Create a function, `PlotPower()`, that allows you to create a plot of x against x^a for a fixed a and a sequence of values of x. For instance, if you call

```
PlotPower(np.arange(1, 11),3)
```

then a plot should be created with an x-axis taking on values 1, 2, . . . , 10, and a y-axis taking on values 1^3,2^3 , . . . ,10^3.

```
def PlotPower(X, a):
  if X is None or a is None:
    return None
  Y = Power3(X, a)
  fig, ax = plt.subplots(1)
  ax.plot(X,Y)
  ax.set_title(f"PlotPower ({a})")

PlotPower(np.arange(1,11), 3)
```




```
allDone();
```

```
<IPython.lib.display.Audio object>
```