The `continue` statement and the `break` statement

This notebook introduces you to the `continue` statement and the `break` statement. Under normal circumstances the while loop will execute as long as the conditional expression is True. However, sometimes, we want to stop executing the loop based on some intermediate state of the program. In that case, to exit the loop, we use the `break` statement. When program execution encounters a break statement, it will terminate the loop and continue with code outside the loop. In other cases, we wish just the current iteration of the loop to be terminated and want execution to continue with the next iteration of the loop. In such situations, we use the `continue` statement.

In [1]:

```
'''
This example demonstrates the use of the 'continue' statement.
Print the squares of all integers between 1 and 10 except 7,
'''

n = 1
while n <= 10:
    if n == 7:
        n += 1
        continue #Use the continue statement to skip this iteration
    print(n**2)
    n += 1
```

```
1
4
9
16
25
36
64
81
100
```

In [2]:

```
'''
This example demonstrates the use of the 'break' statement.
In this code, we ask the user to enter a positive integer.  We then print the square of the intege
r.  We do this
as long as the user enters any positive value except 9999.  If the user enters 9999, we stop.
But suppose, the user enters a negative value or 0, we terminate the loop using the 'break' statem
ent.
'''
n = int(input('Enter a positive integer.  Enter 9999 to end: '))
while n != 9999:
    if n < 1:
        print('You entered a non-positive value.  We are done!!')
        break
    print(n**2)
    n = int(input('Enter a positive integer.  Enter 9999 to end: '))
print('Done')
```

```
Enter a positive integer.  Enter 9999 to end: 10
100
Enter a positive integer.  Enter 9999 to end: -2
You entered a non-positive value.  We are done!!
Done
```

In [3]:

```
'''
This is another example of the use of the 'break' in a while loop.
The while loop will execute as long as the user enters any value but 'done'.
This example is just to demonstrate the use of the break statement.  Note that we could have writt
en this code
using just a normal while loop.
Also notice the conditional expression in the while loop.  This expression will always be True. Th
```

```
erefore, if we did
not have the 'break' statement inside the loop, we would get an infinite loop.
'''
while True:
    line = input("enter a string.  Enter 'done' to end: ")
    if line == 'done':
        break
    print(line)
```

```
enter a string.  Enter 'done' to end: Hi there
Hi there
enter a string.  Enter 'done' to end: Yes
Yes
enter a string.  Enter 'done' to end: done
```