In [1]:
```
# Q1
spam = 100
# Remember, if you want it to throw error if it is less than 10
# then you have to test if it is greater than 10.
# Less than 10 is the error condition, not the expression for assert.
assert spam >= 10, 'Your spam is less than 10!'
```

In [2]:
```
# Q2
eggs = 'hello'
bacon = 'good bye'

# Raise an AssertError if they are not different.
assert eggs.lower() != bacon.lower(), 'eggs/bacon should not be the same!'
```

In [3]:
```
# Q3 assert True, 'Always triggers an AssertionError.'
```

In [ ]:
```
 # Q4
#   Test a logging.debug('message')
#    2019-08-03 12:24:50,549 - DEBUG - This is a test message.
#   logging.debug('This is a test message.')
```

In [4]:
```
# Q5
import logging
logging.basicConfig(
    filename='programLog.txt',
    level=logging.DEBUG,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
```

In [5]:
```
# Q6
# logging.debug() - variable's state and small details
# logging.info() - general events, confirm a program is working
# logging.warning() - potiential problem to work on in the future
# logging.error() - record an error that caused program to fail to do something
# logging.critical() - fatal error that has caused
```

In [6]:
```
# Q7 logging.disable(logging.DEBUG)
```

In [7]:
```
#Q8 # Because with print, when your program is ready for production, you still
#     have to "remove" or comment it out. Verses logging message, you can toggle
#     the setting on/off or write to a file (send to a server). It is more flexible
#     especially with logging level 1-5.
```

In [8]:
```
# Q9  Step - one line execution at a time
# Over - excecute the next line of code, but if it is a program, it will
#        complete the entire function call.
# out  - execute the lines of code unti it returns from the current function.
#        (out is useful when you stepped into a function call).
```

In [9]:
```
# Q10  Go runs until the program terminate or reaches a breakpoint set.
```

In [ ]:
```
# Q11 #  When you have Debugger enabled and you can right click on any lines
#     to create a breakpoint. During Go - it will stop there and await your next
#    command.
```