(1) Namespaces: Global vs. Local Variables

In [ ]:
```
'''
Q1: Let us assume that you've defined a function triple() as shown below:

def triple(a):
    b = 3
    print('a = {}, b = {}'.format(a, b))
    return a * b

In you function, a variable 'b' is local variable.

Rewrite (i.e. redefine) the function triple() in which a variable 'b' is a global varia
'''
```

In [1]:
```
b = 3
def triple(a):

    print('a = {}, b = {}'.format(a, b))
    return a * b
```

In [2]:
```
triple(1)
```

```
a = 1, b = 3
```

Out[2]:
```
3
```

(2) Exception Handling

In [ ]:
```
'''
Q2: Let us assume that you defined a function open_new() as shown below:

def open_new(filename, mode):
    infile = open(filename, mode)
    contents = infile.read().split()
    infile.close()
    return contents

When the file doesn't exist in the current working directory,
running the open_new() function will trigger IOError.

Write a new function safe_open_new() that takes filename and mode as input arguments.

If an exception (i.e. errors) is rased while trying to open the file,
your newly defined safe_open_new() will return the following message, not the ordanary

'Master, I cannot find your treasure in the working directory...'
'''
```

In [3]:
```
def safe_open_new(filename, mode):
    try:
```

```
        infile = open(filename, mode)
        contents = infile.read().split()
        infile.close()
        return contents
    except:
        print('Master, I cannot find your treasure in the working directory...')
```

In [4]:
```
open_new('randomname.txt', 'r')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-fb3fc89673a8> in <module>
----> 1 open_new('randomname.txt', 'r')

NameError: name 'open_new' is not defined
```

In [5]:
```
safe_open_new('randomname.txt', 'r')
```

```
Master, I cannot find your treasure in the working directory...
```

In [ ]:
```
'''
Q3: Let us assume that you defined a function math_game() as shown below:

def math_game(n):
    import random
    count = 0
    for i in range(n):
        x = random.randrange(0, 10)
        y = random.randrange(0, 10)
        numbers = x + y
        print ('{} + {} ='.format(x, y))
        guess = int(input('Enter your guess: '))
        if guess == numbers:
            count += 1
            print("Correct.")
        elif guess != numbers:
            print("Incorrect.")
    print('You got {} correct answer(s) out of {}.'.format(count, n))

One sample run of math_game() is following:

>>> math_game(2)
3 + 1 =
Enter your guess: 4
Correct.
4 + 9 =
Enter your guess: 11
Incorrect.
You got 1 correct answer(s) out of 2.

However, this function math_game() requires users to to enter their answers using digit

Write a new function new_math_game() that:

Takes a number of rounds as an input. The function can handle non-digit user input by:
(1) Printing a message like "Please write your answer using digits 0 through 9. Try aga
(2) Then, give the user another opportunity to enter an answer.
```

```
One sample run of new_math_game() is following:

>>> new_math_game(3)
4 + 3 =
Enter your guess: 7
Correct.
7 + 8 =
Enter your guess: fifteen
Please write your answer using digits 0 through 9. Try again!
Enter your guess: 15
Correct.
2 + 8 =
Enter your guess: 9
Incorrect.
You got 2 correct answer(s) out of 3.
'''
```

In [1]:

```
def new_math_game(n):
    while True:
        try:
            import random
            count = 0
            for i in range(n):
                x = random.randrange(0, 10)
                y = random.randrange(0, 10)
                numbers = x + y
                print ('{} + {} ='.format(x, y))
                guess = int(input('Enter your guess: '))
                if guess == numbers:
                    count += 1
                    print("Correct.")
                elif guess != numbers:
                    print("Incorrect.")
            print('You got {} correct answer(s) out of {}.'.format(count, n))
        except:
            print('Please write your answer using digits 0 through 9. Try again!')
```

In [3]:

```
new_math_game(3)
```

```
4 + 3 =
Enter your guess: 7
Correct.
7 + 8 =
Enter your guess: fifteen
Please write your answer using digits 0 through 9. Try again!
Enter your guess: 15
Correct.
2 + 8 =
Enter your guess: 9
Incorrect.
You got 2 correct answer(s) out of 3.
```

In [ ]:

```
'''
Q4: In Q3, you defined a function new_math_game().

Your function gives another opportunity to enter an answer.

What if you wish to limit the number of opportunities when users can input non-digits?
```

```
Write a new function new_math_game_2() that:

Takes a number of rounds as an input. The function can handle non-digit user input by:
(1) Printing a message like "Please write your answer using digits 0 through 9. Try aga
(2) Them give the user another opportunity to enter an answer; but
(3) When users input non-digits more than two times, the function will quit and print f
    'Two erros. Quitting...'

Two sample runs of new_math_game_2() is following:

>>> new_math_game_2(3)
4 + 3 =
Enter your guess: 7
Correct.
7 + 8 =
Enter your guess: fifteen
Please write your answer using digits 0 through 9. Try again!
Enter your guess: 15
Correct.
2 + 8 =
Enter your guess: 9
Incorrect.
You got 2 correct answer(s) out of 3.

>>> new_math_game_2(3)
4 + 3 =
Enter your guess: 7
Correct.
7 + 8 =
Enter your guess: fifteen
Please write your answer using digits 0 through 9. Try again!
Enter your guess: 15
Correct.
2 + 8 =
Enter your guess: ten
Two errors. Quitting...
'''
```

In [1]:

```python
def new_math_game_2(n):
    while True:
        try:
            import random
            count = 0
            for i in range(n):
                x = random.randrange(0, 10)
                y = random.randrange(0, 10)
                numbers = x + y
                print ('{} + {} ='.format(x, y))
                guess = int(input('Enter your guess: '))
                if guess == numbers:
                    count += 1
                    print("Correct.")
                elif guess != numbers:
                    print("Incorrect.")
            print('You got {} correct answer(s) out of {}.'.format(count, n))
        except:
            print('Please write your answer using digits 0 through 9. Try again!')
            print('Two errors. Quitting...')
```

In [1]:
```
new_math_game_2(3)
```

```
4 + 3 =
Enter your guess: 7
Correct.
7 + 8 =
Enter your guess: fifteen
Please write your answer using digits 0 through 9. Try again!
Enter your guess: 15
Correct.
2 + 8 =
Enter your guess: ten
Two errors. Quitting...
```

(3) Modules (Top-Level Module)

In [ ]:
```
'''
Q5 and Q6: Python has its own search path, but your desktop folder wouldn't be included
'''
```

In [ ]:
```
# Q5: Print out the search path
```

In [ ]:
```
import sys
sys.path
```

In [ ]:
```
# Q6: Add the desktop folder to the search path
```

In [ ]:
```
import sys
sys.path.append('C:/Users/Desktop')
sys.path
```

In [ ]:
```
'''
Q7 ~ Q8: In Python, you can import math module.
'''
```

In [ ]:
```
# Q7: Display the name of math module
```

In [4]:
```
import math
dir(math)
```

Out[4]:
```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
```

```
'asin',
'asinh',
'atan',
'atan2',
'atanh',
'ceil',
'comb',
'copysign',
'cos',
'cosh',
'degrees',
'dist',
'e',
'erf',
'erfc',
'exp',
'expm1',
'fabs',
'factorial',
'floor',
'fmod',
'frexp',
'fsum',
'gamma',
'gcd',
'hypot',
'inf',
'isclose',
'isfinite',
'isinf',
'isnan',
'isqrt',
'ldexp',
'lgamma',
'log',
'log10',
'log1p',
'log2',
'modf',
'nan',
'perm',
'pi',
'pow',
'prod',
'radians',
'remainder',
'sin',
'sinh',
'sqrt',
'tan',
'tanh',
'tau',
'trunc']
```

In [5]:
```
import math
math.__name__
```

Out[5]:
```
'math'
```

In [ ]:
```
# Q8: Display the absolute pathname of the file containing math module
```

In [ ]:
```
# Attribute Error Due to Windows OS Issue

import math
math.__file__
```

(4) Method Invocations

In [ ]:
```
'''
Q9 ~ Q10: Rewrite the Python codes below with the method invocations.

Example:

s = 'clemson tigers'
s.upper()

str.upper(s)
'''
```

In [10]:
```
s = 'clemson tigers'
lst = ['apple', 'pear', 'strawberry']
```

In [ ]:
```
# Q9: lst.append('blueberry')
```

In [11]:
```
list.append(lst, 'blueberry')
```

In [12]:
```
lst
```

Out[12]:
```
['apple', 'pear', 'strawberry', 'blueberry']
```

In [ ]:
```
# Q10: s.upper().split()
```

In [13]:
```
str.upper(s).split()
```

Out[13]:
```
['CLEMSON', 'TIGERS']
```