-- assert and try/except

#DEMO: assert
#==========================================================
#Check type of argument
def btod(bin_str):
    assert isinstance(bin_str, str)
    print("Your string is " + bin_str)
    return None

btod('1100101')
btod(1100101)

#==========================================================
#Check type of arg as in solution 4-2 - optional msg
def btod(bin_str):
    for bit in bin_str:
        assert bit in '01', 'A binary number is needed.'
        print("Your string is " + bin_str)
    return None

btod('1100101')
btod('123')

#==========================================================
#check that x and y are integers and y is greater than 0
def foo(x, y):
    assert isinstance(x, int)
    assert isinstance(y, int)
    assert y > 0

    return x // y

print(foo(8, 4))
print(foo(8.0, 4.0))
print(foo(8, 0))

#==========================================================
#DEMO: try/except
#===================================================================================
# Example: Catch ANY exception
def catch_them_all():
    try:
        x = 1/0  # <--- obvious division by 0 error
    except:
        print("Caught an exception")

#===================================================================================
# Example: Catch ONE specific exception
def catch_divZero():
    try:
        x = 1/0
    except ZeroDivisionError:
        print("Caught a ZeroDivisionError")

#===================================================================================

# Exercise: assert

#Write a function, double_your_num, that asks the user
#to enter an integer between 1 and 10, and then
#prints the number * 2.
#Check that the input value is valid (type and range).
#Test your code with correct and incorrect input values.
# Exercises: try/except

# Given the following Python code
# to increment the first n integers
# in a list of integers by one:

# Then revise the function to include
# try/except to implement the
# second example in the docstring.

#==========================================================
Generate a ZeroDivisionError:

Write Python code to catch a zero
division error and print a msg:
Denominator cannot be 0.

Generate a KeyError:

Write Python code to catch a key
error and print a message:
<incorrect value> is not a key.

Generate a FileNotFoundError:

Write Python code to catch a
file not found error and print
a message:
Where is <filename>??
# Write a new version of double_your_num,
# but for an invalid input value, print an
# error message and ask the user to re-enter
# the number, until a valid value is entered.

#===============================================
#===============================================
