SOFTWARE QUALITY

Assignment 2

Edward King
100553812
March 7th 2017

Test Automation
Conversion Program

The software that is being tested is the same created in Assignment 1, which is the conversion program to convert between base 2 number and base 10. The test cases that are being automated are many of the manual test cases used in assignment 1. The test case framework that is being used is PyUnit, used as unittest in the python code, and is a standard package with python.

Automated Test Cases

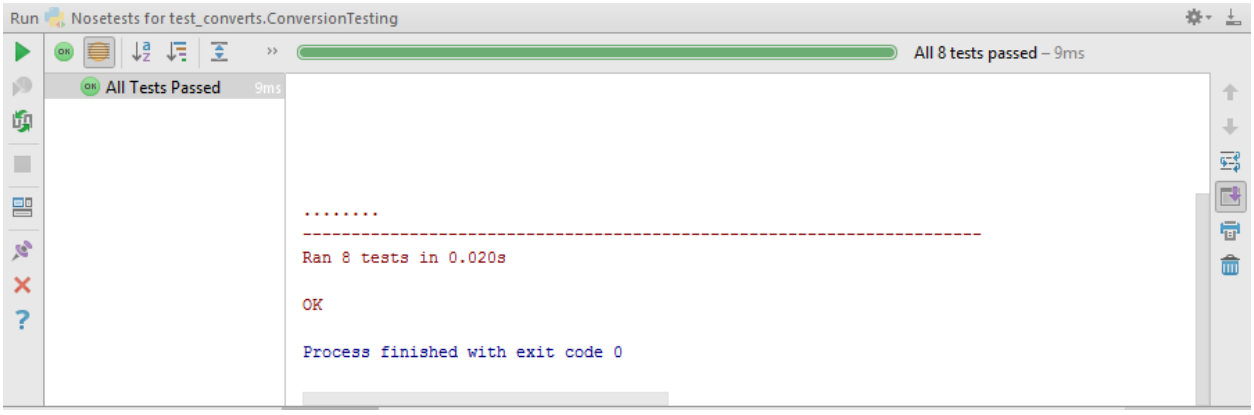
A total of 6 test cases were created from the manual test cases used in assignment, shown below.

```
from unittest import TestCase
import Conversions


class ConversionTesting(TestCase):
    def test_DecToBinZero(self):
        self.assertEqual(Conversions.ConvertDecToBin.DecToBin(0), 0)

    def test_DecToDinOne(self):
        self.assertEqual(Conversions.ConvertDecToBin.DecToBin(1), 1)

    def test_DecToBin42(self):
        self.assertEqual(Conversions.ConvertDecToBin.DecToBin(42), 101010)

    def test_DecToBinLarge(self):
        self.assertEqual(Conversions.ConvertDecToBin.DecToBin(512), 1000000000)

    def test_BinToDecZero(self):
        self.assertEqual(Conversions.ConvertBinToDec.BinToDec(0), 0)

    def test_BinToDecOne(self):
        self.assertEqual(Conversions.ConvertBinToDec.BinToDec(1), 1)

    def test_BinToDec42(self):
        self.assertEqual(Conversions.ConvertBinToDec.BinToDec(101010), 42)

    def test_BinToDecLarge(self):
        self.assertEqual(Conversions.ConvertBinToDec.BinToDec(1000000000), 512)
```

The test cases are split up in two groups, Decimal to Binary, then Binary to Decimal. Four main tests were done on each group, the edge case 0, a value of one, a random number, and a large number to test the conversion speed.

Running the test cases

As I am using python, my IDE of choice is PyCharm. In PyCharm, it has a simple GUI to run test cases the user has programed, below is the result of running the tests.



As shown, all the test cases passed with no errors or failures.

Challenges

The biggest challenged faced when creating the testing was learning how to use the PyUnit testing framework, as it is slightly different then JUnit done in class. Another challenge was applying the principles of unit tests, to provide good coverage for the class and not try to test a large amount of numbers, creating unnecessary test cases. The menu system could not be properly tested however because of the way it was coded, and because of it, the error checking could not be tested with PyUnit. Knowing this for the future, proper coding practices can be followed to allow testing of error checking and code like a menu system.

Implementation

Code used in the program

Decimal to Binary

```
def DecToBin(x):  # Decimal to Binary conversion
    k = []  # Initialing the list used in the conversions
    if x == 0:  # Checks if the inputted number is 0, it's a special case for the program
        k.append(0)  # If the input is 0, the output will be 0
    while (x > 0):  # Loops until x is 0
        a = int((x % 2))  # Checking if the binary is 1 or 0
        k.append(a)  # Adding the number to a list to take out later
        x = (x - a) / 2  # Moving to the next number
    out = ""  # Defining the out variable
    for j in k[::-1]:  # For loop for the list k, taking all the binary numbers that was converted
        out = out + str(j)  # Adding them to "out"
    out = int(out)
    return out  # giving back the new binary number.
```

Binary to Decimal

```
def BinToDec(x):  # Binary to Decimal conversion
    out = 0  # Setting out to 0, special case incase of user inputting 0 to convert.
    j = 1  # Binary conversions start at 1
    while (x >= 1):  # While loop, keeps looping until no more numbers
        a = (x % 10) # Divides by 10 and gets the remainder ( 1 or 0 )
        if a == 1:  # Checking if the binary digit is one
            out = out + j  # Adds the current place to the output
        elif a == 0: # Checks if the binary digit is 0, does nothing
            out = out
        else: # If it's anything else, it is not a binary number.
            print("Please input a valid binary number ( only 1's or 0's. )")
            out = -1
            return out
        j = j + j  # Goes to the next place in the binary conversions
        x = int(x / 10) # Gets rid of another number
    return out # Returns the converted number
```

Built in convertors, used for checking programmed conversions

```
def DecToBinCheck(x):  # Built in converting, used to check if the number was correct
    return bin(x)


def BinToDecCheck(x):  # Built in converting, used to check if the number was correct
    return int(x, 2)
```

Main Function

```
if __name__ == '__main__':
    DTB = Conversions.ConvertDecToBin
    BTD = Conversions.ConvertBinToDec
    conversion = "na"  # No input at the start, needed for the while loop.
    while conversion.upper() != "X":  # Exits the program if input is X
        conversion = input("What would you like to convert from? \n D - Decimal to Binary \n B - Binary to Decimal "
                           "\n X - Exit\n")  # First part of the menu, asking for
        if conversion.upper() == "D":  # Decimal to Binary
            number = (input("Type your decimal number to convert to binary\n")) # Asking user for input
            if number.isdigit(): # Checks if the number is a valid integer
                if int(number) > 10000000000000000000000:  # Tells user not to input too high of an number
                    print("Please enter a number less than 100000000000000000000000.")  # Ints are used to store the num
                else:
                    x = int(number)
                    convert = DTB.DecToBin(x) # Converts the number
                    print("Your converted number is: %s" % convert)  # Displays answer
                    # Built in converting, used to check if the number was correct
                    print("The built in converter converts it to: %s" % DecToBinCheck(int(number)))
            else:  # Number is not valid
                print("Please enter a valid number.")
        elif conversion.upper() == "B":  # Binary to Decimal
            number = (input("Type your binary number to convert to decimal\n"))  # Asking user for input
            if number.isdigit():  # Checks if the number is valid
                if int(number) > 10000000000000000000000:  # Tells user not to input too high of an number
                    print("Please enter a number less than 100000000000000000000000.")  # Ints are used to store the num
                else:
                    x = int(number)
                    convert = BTD.BinToDec(x)  # Converts the number
                    if convert == -1: # Number was not binary
                        pass
                    else:  # Number is binary
                        print("Your converted number is: %s" % convert)  # Prints output
                        # Built in converting, used to check if the number was correct
                        print("The built in converter converts it to: %s" % BinToDecCheck(number))
            else:  # Number was not a valid number
                print("Please enter a valid number.")
        elif conversion.upper() == "X":  # Exiting the program
            exit(0)
        else:  # Argument not valid for the menu
            print("Please input a valid argument.")
    exit(0)  # Exiting program incase for whatever reason the exit button does not work
```