Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Guadalajara

Maestría en Ciencias de la Computación

TC4002.1 Análisis, diseño y construcción de software
Dr. Gerardo Padilla Zarate

Nombre: Héctor Gabriel Olagues Torres
Matrícula: A00354877

Abril de 2021

Lab 8.1 – Logging and Debugging

Goals: This exercise provides the experience to implement different logging configurations for your application. As well as to try to debug issues in two software programs.

Context: You are requested to use the Python logging module for such practice.

Request:

1. You have been asked to triage (find the issue of a program)
    a. The program is included in the file **BouncingBalls_bug.py**
    b. Failures
        i. The program is not behaving as it was specified, it should keep running with the bouncing ball for endless time until the user closes the application
        ii. The speed of the bouncing ball should be the same all the time
        iii. When you click the space bar another ball should appear
    c. The programs has three injected defects
    d. Use any debugger looking to use features such as
        i. Exploring the value/state of variables/objects
        ii. Stop execution using breakpoints
        iii. Any other tool feature that may help you
    e. Upload your programs including two versions in your GIT repo – the one with the fixes and the one with the failures

Evidence:

First issue – The program keeps running for endless time by using same change value for both X and Y.

Before

```
# Bounce the ball if needed
if ball.y > SCREEN_HEIGHT - BALL_SIZE or ball.y < BALL_SIZE:
    ball.change_y *= -2
if ball.x > SCREEN_WIDTH - BALL_SIZE or ball.x <= BALL_SIZE:
    ball.change_x *= -1
```

After

```
# Bounce the ball if needed
if ball.y > SCREEN_HEIGHT - BALL_SIZE or ball.y < BALL_SIZE:
    ball.change_y *= -1
if ball.x > SCREEN_WIDTH - BALL_SIZE or ball.x < BALL_SIZE:
    ball.change_x *= -1
```

Second issue – Fixed the speed of the bouncing ball to be the same.

Before

```
# Bounce the ball if needed
if ball.y > SCREEN_HEIGHT - BALL_SIZE or ball.y < BALL_SIZE:
    ball.change_y *= -2
if ball.x > SCREEN_WIDTH - BALL_SIZE or ball.x <= BALL_SIZE:
    ball.change_x *= -1
```

After

```
# Bounce the ball if needed
if ball.y > SCREEN_HEIGHT - BALL_SIZE or ball.y < BALL_SIZE:
    ball.change_y *= -1
if ball.x > SCREEN_WIDTH - BALL_SIZE or ball.x < BALL_SIZE:
    ball.change_x *= -1
```

Third issue – When clicking the space bar only one ball appears, instead of two.

Before

```
# Space bar! Spawn a new ball.
if event.key == pygame.K_SPACE:
    ball = make_ball()
    ball_list.append(ball)
    ball = make_ball()
    ball_list.append(ball)
```

After

```
# Space bar! Spawn a new ball.
if event.key == pygame.K_SPACE:
    ball = make_ball()
    ball_list.append(ball)
```

For debugging I used embedded debugger from PyCharm.

2. You have been asked to triage (find the issue of a program)
    a. The program is included in the file **tictactoe_bug.py**
    b. Failures
        i. The program is not behaving as it was specified
    c. The programs has two injected defects
    d. Use any debugger looking to use features such as
        i. Exploring the value/state of variables/objects
        ii. Stop execution using breakpoints
        iii. Any other tool feature that may help you
    e. Upload your programs including two versions in your GIT repo – the one with the fixes and the one with the failures

Evidence:

First issue – Fixed the position 5 and 6 of the board in the middle.

Before

```
def printBoard(board):
    print(board['7'] + '|' + board['8'] + '|' + board['9'])
    print('-+-+-')
    print(board['4'] + '|' + board['6'] + '|' + board['5'])
    print('-+-+-')
    print(board['1'] + '|' + board['2'] + '|' + board['3'])
```

After

```
def printBoard(board):
    print(board['7'] + '|' + board['8'] + '|' + board['9'])
    print('-+-+-')
    print(board['4'] + '|' + board['5'] + '|' + board['6'])
    print('-+-+-')
    print(board['1'] + '|' + board['2'] + '|' + board['3'])
```

Second issue – Fixed the condition for winning, from down to top in the middle.

Before

```
    elif theBoard['2'] == theBoard['5'] == theBoard['7'] != ' ': # down the middle
        printBoard(theBoard)
        print("\nGame Over.\n")
        print(" **** " +turn + " won. ****")
        break
```

After

```
elif theBoard['2'] == theBoard['5'] == theBoard['8'] != ' ': # down the middle
    printBoard(theBoard)
    print("\nGame Over.\n")
    print(" **** " +turn + " won. ****")
    break
```

For debugging I used embedded debugger from PyCharm.

3. Add logs for one program that managed a directory (Lab 3.2)
    a. Add logs for INFO messages
    b. Add logs for ERROR messages
        i. Configure to send the log information to the console
    c. Create this new lab as a new program in your repo
    d. Save one run screenshot showing the logs and upload it to Canvas also

Evidence:

```
test_add_record (test_records_handler.TestRecordsHandler) ...
INFO:Creating connection

INFO:Table creation

INFO:Deleting all the records

INFO:Listing all the records

INFO:Adding record

INFO:Listing all the records

INFO:The connection is closed
ok
test_delete_record (test_records_handler.TestRecordsHandler) ...
INFO:Creating connection

INFO:Table creation

INFO:Deleting all the records

INFO:Listing all the records

INFO:Adding record

INFO:Adding record

INFO:Adding record

INFO:Listing all the records

INFO:Deleting record

INFO:Listing all the records

INFO:The connection is closed
ok
```

```
test_list_all (test_records_handler.TestRecordsHandler) ...
INFO:Creating connection

INFO:Table creation

INFO:Deleting all the records

INFO:Adding record

INFO:Listing all the records

INFO:The connection is closed
ok
test_look (test_records_handler.TestRecordsHandler) ...
INFO:Creating connection

INFO:Table creation

INFO:Deleting all the records

INFO:Looking for a specific record

INFO:Adding record

INFO:Looking for a specific record

INFO:The connection is closed
ok

----------------------------------------------------------------------
Ran 12 tests in 0.045s

OK
```

Only logging INFO messages are displayed in the console. No logging ERROR messages were displayed, as all erroneous conditions are reached only when the SQL functions are not executed accordingly. Nevertheless, all logging ERROR messages were added in all exception sentences.

```
except Error as e:
    print(e)
    logging.basicConfig(format='\n%(levelname)s:%(message)s', level=logging.INFO)
    logging.error("Connection not established with Database")

except Error as e:
    print(e)
    logging.basicConfig(format='\n%(levelname)s:%(message)s', level=logging.INFO)
    logging.error("Table was not created")

except Error as e:
    print(e)
    logging.basicConfig(format='\n%(levelname)s:%(message)s', level=logging.INFO)
    logging.error("The record was not added")

except Error as e:
    print(e)
    logging.basicConfig(format='\n%(levelname)s:%(message)s', level=logging.INFO)
    logging.error("The record was not deleted")

except Error as e:
    print(e)
    logging.basicConfig(format='\n%(levelname)s:%(message)s', level=logging.INFO)
    logging.error("Not able to find the record")

except Error as e:
    print(e)
    logging.basicConfig(format='\n%(levelname)s:%(message)s', level=logging.INFO)
    logging.error("Not able to list all the records")

except Error as e:
    print(e)
    logging.basicConfig(format='\n%(levelname)s:%(message)s', level=logging.INFO)
    logging.error("Not able to delete all the records")
```

Link to GitHub Repo

https://github.com/hectorolagues/TC4002_dev_exercises

https://github.com/hectorolagues/TC4002_dev_exercises/tree/main/Lab8.1