```
#-----Statement of Authorship----------------------------------------#
#
#  This is an individual assessment task for QUT's teaching unit
#  IFB104, "Building IT Systems", Semester 1, 2023.  By submitting
#  this code I agree that it represents my own work.  I am aware of
#  the University rule that a student must not act in a manner
#  which constitutes academic dishonesty as stated and explained
#  in QUT's Manual of Policies and Procedures, Section C/5.3
#  "Academic Integrity" and Section E/2.1 "Student Code of Conduct".
#
#  Put your student number here as an integer and your name as a
#  character string:
#
student_number = 10583122
student_name   = 'Sharyn Tauro'
#
#  NB: Files submitted without a completed copy of this statement
#  will not be marked.  All files submitted will be subjected to
#  software plagiarism analysis using the MoSS system
#  (http://theory.stanford.edu/~aiken/moss/).
#
#--------------------------------------------------------------------#



#-----Assessment Task 1 Description----------------------------------#
#
#  This assessment task tests your skills at processing large data
#  sets, creating reusable code and following instructions to display
#  a complex visual image.  The incomplete Python program below is
#  missing a crucial function.  You are required to complete this
#  function so that when the program runs it fills a grid with various
#  symbols, using data stored in a list to determine which symbols to
#  draw and where.  See the online video instructions for
#  full details.
#
#  Note that this assessable assignment is in multiple parts,
#  simulating incremental release of instructions by a paying
#  "client".  This single template file will be used for all parts
#  and you will submit your final solution as this single Python 3
#  file only, whether or not you complete all requirements for the
#  assignment.
#
#  This file relies on other Python modules but all of your code
#  must appear in this file only.  You may not change any of the code
#  in the other modules and you should not submit the other modules
#  with your solution.  The markers will use their own copies of the
#  other modules to test your code, so your solution will not work
#  if it relies on changes made to any other files.
#
#--------------------------------------------------------------------#



#-----Preamble-------------------------------------------------------#
#
# This section imports necessary functions used to execute your code.
# You must NOT change any of the code in this section, and you may
```

```
# NOT import any non-standard Python modules that need to be
# downloaded and installed separately.
#

# Import standard Python modules needed to complete this assignment.
# You should not need to use any other modules for your solution.
# In particular, your solution must NOT rely on any non-standard
# Python modules that need to be downloaded and installed separately,
# because the markers will not have access to such modules.
from turtle import *
from math import *
from random import *
from sys import exit as abort
from os.path import isfile

# Confirm that the student has declared their authorship
if not isinstance(student_number, int):
    print('\nUnable to run: No student number supplied',
          '(must be an integer), aborting!\n')
    abort()
if not isinstance(student_name, str):
    print('\nUnable to run: No student name supplied',
          '(must be a character string), aborting!\n')
    abort()

# Import the functions for setting up the drawing canvas
if isfile('assignment_1_config.py'):
    print('\nConfiguration module found ...\n')
    from assignment_1_config import *
else:
    print("\nCannot find file 'assignment_1_config.py', aborting!\n")
    abort()

# Define the function for generating data sets, using the
# imported raw data generation function if available, but
# otherwise creating a dummy function that just returns an
# empty list
if isfile('assignment_1_data_source.py'):
    print('Data generation module found ...\n')
    from assignment_1_data_source import raw_data
    def data_set(new_seed = randint(0, 99999)):
        print('Using random number seed', new_seed, '...\n')
        seed(new_seed) # set the seed
        return raw_data() # return the random data set
else:
    print('No data generation module available ...\n')
    def data_set(dummy_parameter = None):
        return []

#
#-----------------------------------------------------------------#



#-----Student's Solution-------------------------------------------#
#
#  Complete the assignment by replacing the dummy function below with
#  your own function and any other functions needed to support it.
#  All of your solution code must appear in this section.  Do NOT put
```

```
#  any of your code in any other sections and do NOT change any of
#  the provided code except as allowed by the comments in the next
#  section.
#

# All of your code goes in, or is called from, this function.
# Make sure that your code does NOT call function data_set (or
# raw_data) because it's already called in the main program below.
def visualise_data(rename_me):


    #### Function to draw the 6 Hexagons

    ''' @param x_1 and y_1 takes the co-ordinates on the canvas where the turtle
needs to be positioned before starting to draw the hexagons
    '''

    def drawHexagon(x_1:int,y_1:int):
        goto(x_1,y_1)
        pencolor("grey")
        fillcolor("BlanchedAlmond")
        width(2)
        pendown()
        begin_fill()
        for i in range(6):
            forward(60)
            right(60)
        end_fill()
        penup()



    #### Function to draw the Oblong Body of the Owl

    def drawOblong_brown():
        forward(67)
        pendown
        pensize(1)
        pencolor("brown4")
        fillcolor("brown4")
        begin_fill()
        circle(30,180)
        forward(30)
        circle(30,180)
        forward(40)
        end_fill()


    #### Function to draw the Golden Circles of the Owl eyes

    def drawCircle_golden():
        penup()
        backward(30)
        left(50)
        pencolor('DarkGoldenrod1')
        fillcolor('DarkGoldenrod1')
        begin_fill()
        circle(22)
        end_fill()
```

```
        left(33)
        forward(47)
        begin_fill()
        circle(22)
        end_fill()


#### Function to draw the Large White Circles of the Owl eyes

    def drawCircle_white_1():
        forward(12)
        left(90)
        forward(22)
        pendown()
        pencolor('white')
        fillcolor('white')
        begin_fill()
        circle(15)
        end_fill()
        penup()
        left(64)
        forward(52.5)
        left(90)
        forward(22)
        pendown()
        begin_fill()
        circle(15)
        end_fill()
        penup()



#### Function to draw the Black Circles of the Owl eyes

    def drawCircle_black():
        left(45)
        forward(10)
        pendown()
        pencolor('black')
        fillcolor('black')
        begin_fill()
        circle(10)
        end_fill()
        penup()
        left(65)
        forward(38)
        pendown()
        begin_fill()
        circle(10)
        end_fill()
        penup()



#### Function to draw the Small White Circles of the Owl eyes

    def drawCircle_white_2():
        left(175)
        forward(27)
```

```
        pendown()
        pencolor('white')
        fillcolor('white')
        begin_fill()
        circle(2)
        end_fill()
        penup()
        right(170)
        forward(27)
        pendown()
        begin_fill()
        circle(2)
        end_fill()
        penup()


#### Function to draw the Orange Triangles of the Owl legs

    def drawTriangle_orange_1():
        right(75)
        forward(52)
        pendown()
        pencolor('orange')
        fillcolor('orange')
        begin_fill()
        forward(15)
        right(120)
        forward(20)
        right(130)
        forward(20)
        end_fill()
        left(170)
        penup()
        forward(52)
        pendown()
        begin_fill()
        right(120)
        forward(18)
        right(120)
        forward(20)
        right(122)
        forward(21)
        end_fill()


#### Function to draw the Orange Triangle of the Owl nose

    def drawTriangle_orange_2():
        right(120)
        penup()
        forward(51)
        right(72)
        forward(25)
        pendown()
        begin_fill()
        right(120)
        forward(18)
```

```
        right(120)
        forward(20)
        right(122)
        forward(21)
        end_fill()



    #### Function to draw the entire Owl


    def drawOwl(config:any):
        penup
        '''    @param x_2 and y_2 takes the co-ordinates on the canvas where the
turtle needs to be positioned before starting to draw the Owl
               @param direction gives the compass angle that the turtle needs to
moved to before starting to draw the Owl inside the Hexagon
               @param x_3 and y_3 takes the co-ordinates on the canvas where the
turtle needs to be positioned before starting to write the Direction the Owl is
facing
        '''
        goto(config['x_2'],config['y_2'])
        left(config['direction'])
        drawOblong_brown()
        drawCircle_golden()
        drawCircle_white_1()
        drawCircle_black()
        drawCircle_white_2()
        drawTriangle_orange_1()
        drawTriangle_orange_2()
        up()
        ht()
        goto(config['x_3'],config['y_3'])
        st()
        down()
        width(1)
        pencolor("grey")
        write(config['Text'],align = 'center', font = ('Arial', 23, 'normal'))
        penup()




### Calling the function drawHexagon to draw the 6 Hexagons

    drawHexagon(-550,260) # Hexagon 1
    drawHexagon(-550,50)  # Hexagon 2
    drawHexagon(-550,-160)# Hexagon 3
    drawHexagon(500,260)  # Hexagon 4
    drawHexagon(500,50)   # Hexagon 5
    drawHexagon(500,-160) # Hexagon 6




### Drawing Owl 1

    Owl_1: any = {"x_2":-550,"y_2":260,
```

```
                "direction":270,
                "x_3":-520, "y_3":110,
                "Text":"North"}
    drawOwl(Owl_1)



### Drawing Owl 2

    Owl_2: any = {"x_2":-492,"y_2":52,
                "direction":213,
                "x_3":-520, "y_3":-100,
                "Text":"North East"}
    drawOwl(Owl_2)



### Drawing Owl 3

    Owl_3: any = {"x_2":-458,"y_2":-207,
                "direction":210,
                "x_3":-520, "y_3":-310,
                "Text":"South East"}
    drawOwl(Owl_3)



### Drawing Owl 4

    Owl_4: any = {"x_2":563,"y_2":156,
                "direction":205,
                "x_3":540, "y_3":110,
                "Text":"South"}
    drawOwl(Owl_4)



### Drawing Owl 5

    Owl_5: any = {"x_2":501,"y_2":-58,
                "direction":210,
                "x_3":540, "y_3":-95.5,
                "Text":"South West"}
    drawOwl(Owl_5)



### Drawing Owl 6

    Owl_6: any = {"x_2":470,"y_2":-210,
                "direction":205,
                "x_3":540, "y_3":-310,
                "Text":"North West"}
    drawOwl(Owl_6)
```

```
#-----------------------------------------------------------------#



#-----Main Program to Run Student's Solution-------------------------#
#
# This main program configures the drawing canvas, calls the student's
# function and closes the canvas.  Do NOT change any of this code
# except as allowed by the comments below.  Do NOT put any of
# your solution code in this section.
#

# Configure the drawing canvas
# ***** You can change the background and line colours, and choose
# ***** whether or not to draw the grid and other elements, by
# ***** providing arguments to this function call
create_drawing_canvas(canvas_title = "Wise Owl The More You See The Less You Talk",
                      bg_colour = 'light grey',
                      line_colour = 'slate grey',
                      draw_grid = True,
                      write_instructions = False)

# Call the student's function to process the data set
# ***** While developing your program you can call the
# ***** "data_set" function with a fixed seed below for the
# ***** random number generator, but your final solution must
# ***** work with "data_set()" as the function call,
# ***** i.e., for any random data set that can be returned by
# ***** the function when called with no seed
visualise_data(data_set()) # <-- no argument for "data_set" when assessed

# Exit gracefully
# ***** Do not change this function call
release_drawing_canvas(student_name)

#
#-----------------------------------------------------------------#
```