Week 3

March 10, 2021

1 EEE110 Computer Programming Laboratory

1.1 *Kasım Zor and Emre Yorat*

1.2 Week 3

1.2.1 print with Single Quotation Mark

[1]:
```
print('Res. Assist. Emre Yorat')
print('Room 128, North Wing, Prefab Building, Adana Alparslan Türkeş Sci & Tech␣
 ↪University')
print('Sarıçam, Adana, 01250, Turkey')
```

```
Res. Assist. Emre Yorat
Room 128, North Wing, Prefab Building, Adana Alparslan Türkeş Sci & Tech
University
Sarıçam, Adana, 01250, Turkey
```

[2]:
```
print('One
      Two
      Three')
```

```
  File "<ipython-input-2-ae0c4672a876>", line 1
    print('One
          ^
SyntaxError: EOL while scanning string literal
```

1.2.2 print with Double Quotation Mark

[3]:
```
print("Res. Assist. Emre Yorat")
print("Room 128, North Wing, Prefab Building, Adana Alparslan Türkeş Sci & Tech␣
 ↪University")
print("Sarıçam, Adana, 01250, Turkey")
```

Res. Assist. Emre Yorat
Room 128, North Wing, Prefab Building, Adana Alparslan Türkeş Sci & Tech University
Sarıçam, Adana, 01250, Turkey

[4]:
```
print("One
      Two
      Three")
```

```
  File "<ipython-input-4-c312089f618f>", line 1
    print("One
          ^
SyntaxError: EOL while scanning string literal
```

1.2.3 print with Triple Quotation Mark

[5]:
```
print("""I'm reading "Hannibal Rising" tonight.""")
```

```
I'm reading "Hannibal Rising" tonight.
```

[6]:
```
print("""One
Two
Three""")
```

```
One
Two
Three
```

1.2.4 Variable

[7]:
```
# This program demonstrates a variable.
room = 503
print('I am staying in room number')
print(room)
```

```
I am staying in room number
503
```

[8]:
```
# Create two variables: top_speed and distance.
top_speed = 160
distance = 300

# Display the values referenced by the variables.
print('The top speed is')
```

```
print(top_speed)
print('The distance traveled is')
print(distance)
```

```
The top speed is
160
The distance traveled is
300
```

[9]:
```
# This program demonstrates a variable.
room = 503
print('I am staying in room number', room)
```

```
I am staying in room number 503
```

[10]:
```
# This program demonstrates variable reassignment.
# Assign a value to the dollars variable.
dollars = 2.75
print('I have', dollars, 'in my account.')

# Reassign dollars so it references
# a different value.
dollars = 99.95
print('But now I have', dollars, 'in my account!')
```

```
I have 2.75 in my account.
But now I have 99.95 in my account!
```

1.2.5 Storing Strings with the str Data Type

[11]:
```
# Create variables to reference two strings.
first_name = 'Emre'
last_name = 'Yorat'

# Display the values referenced by the variables.
print(first_name, last_name)
```

```
Emre Yorat
```

1.2.6 Reading Input from the Keyboard

[12]:
```
# Get the user's first name.
first_name = input('Enter your first name: ')

# Get the user's last name.
last_name = input('Enter your last name: ')
```

```
# Print a greeting to the user.
print('Hello', first_name, last_name)
```

```
Enter your first name:  Emre
Enter your last name:  Yorat

Hello Emre Yorat
```

1.2.7 Reading Numbers with the input Function

[13]:
```
# Get the user's name, age, and income.
name = input('What is your name? ')
age = int(input('What is your age? '))
income = float(input('What is your income? '))

# Display the data.
print('Here is the data you entered:')
print('Name:', name)
print('Age:', age)
print('Income:', income)
```

```
What is your name?  Emre Yorat
What is your age?  25
What is your income?  7000

Here is the data you entered:
Name: Emre Yorat
Age: 25
Income: 7000.0
```

1.2.8 Performing Calculations

[14]:
```
# Assign a value to the salary variable.
salary = 2500.0

# Assign a value to the bonus variable.
bonus = 1200.0

# Calculate the total pay by adding salary
# and bonus. Assign the result to pay.
pay = salary + bonus

# Display the pay.
print('Your pay is', pay)
```

```
Your pay is 3700.0
```

1.2.9 Calculating a Percentage

[15]:
```
# This program gets an item's original price and
# calculates its sale price, with a 20% discount.

# Get the item's original price.
original_price = float(input("Enter the item's original price: "))

# Calculate the amount of the discount.
discount = original_price * 0.2

# Calculate the sale price.
sale_price = original_price - discount

# Display the sale price.
print('The sale price is', sale_price)
```

```
Enter the item's original price:  100
```

```
The sale price is 80.0
```

1.2.10 Calculating an Average

Determining the average of a group of values is a simple calculation: add all of the values then divide the sum by the number of values. Although this is a straightforward calculation, it is easy to make a mistake when writing a program that calculates an average. For example, let's assume that the variables a, b, and c each hold a value and we want to calculate the average of those values. If we are careless, we might write a statement such as the following to perform the calculation:

average = a + b + c / 3.0

Can you see the error in this statement? When it executes, the division will take place first. The value in c will be divided by 3, then the result will be added to a + b. That is not the correct way to calculate an average. To correct this error, we need to put parentheses around a + b + c, as shown here:

average = (a + b + c) / 3.0

Let's step through the process of writing a program that calculates an average. Suppose you have taken three tests in your computer science class, and you want to write a program that will display the average of the test scores. Here is the algorithm: 1. Get the first test score. 2. Get the second test score. 3. Get the third test score. 4. Calculate the average by adding the three test scores and dividing the sum by 3. 5. Display the average.

In steps 1, 2, and 3 we will prompt the user to enter the three test scores. We will store those test scores in the variables test1, test2, and test3. In step 4, we will calculate the average of the three test scores. We will use the following statement to perform the calculation and store the result in the average variable:

average = (test1 + test2 + test3) / 3.0

Last, in step 5, we display the average.

[16]:
```
# Get three test scores and assign them to the
# test1, test2, and test3 variables.
test1 = float(input('Enter the first test score: '))
test2 = float(input('Enter the second test score: '))
test3 = float(input('Enter the third test score: '))

# Calculate the average of the three scores
# and assign the result to the average variable.
average = (test1 + test2 + test3) / 3.0

# Display the average.
print('The average score is', average)
```

```
Enter the first test score:  20
Enter the second test score:  50
Enter the third test score:  90

The average score is 53.333333333333336
```

1.2.11 The Remainder Operator

[17]:
```
# Get a number of seconds from the user.
total_seconds = float(input('Enter a number of seconds: '))

# Get the number of hours.
hours = total_seconds // 3600

# Get the number of remaining minutes.
minutes = (total_seconds // 60) % 60

# Get the number of remaining seconds.
seconds = total_seconds % 60

# Display the results.
print('Here is the time in hours, minutes, and seconds:')
print('Hours:', hours)
print('Minutes:', minutes)
print('Seconds:', seconds)
```

```
Enter a number of seconds:  10000

Here is the time in hours, minutes, and seconds:
Hours: 2.0
Minutes: 46.0
Seconds: 40.0
```

1.2.12 Converting a Math Formula to a Programming Statement

Suppose you want to deposit a certain amount of money into a savings account and leave it alone to draw interest for the next 10 years. At the end of 10 years, you would like to have $10,000 in the account. How much do you need to deposit today to make that happen? You can use the following formula to find out:

$$P = \frac{F}{(1+r)^n}$$

The terms in the formula are as follows: * P is the present value, or the amount that you need to deposit today. * F is the future value that you want in the account. (In this case, F is $10,000.) * r is the annual interest rate. * n is the number of years that you plan to let the money sit in the account.

In steps 1 through 3, we will prompt the user to enter the specified values. We will assign the desired future value to a variable named future_value, the annual interest rate to a variable named rate, and the number of years to a variable named years. In step 4, we calculate the present value, which is the amount of money that we will have to deposit. We will convert the formula previously shown to the following statement. The statement stores the result of the calculation in the present_value variable.

present_value = future_value / (1.0 + rate)**years

In step 5, we display the value in the present_value variable.

[18]:
```
# Get the desired future value.
future_value = float(input('Enter the desired future value: '))

# Get the annual interest rate.
rate = float(input('Enter the annual interest rate: '))

# Get the number of years that the money will appreciate.
years = int(input('Enter the number of years the money will grow: '))

# Calculate the amount needed to deposit.
present_value = future_value / (1.0 + rate)**years

# Display the amount needed to deposit.
print('You will need to deposit this amount:', present_value)
```

```
Enter the desired future value:  10000
Enter the annual interest rate:  0.18
Enter the number of years the money will grow:  10

You will need to deposit this amount: 1910.6446691360586
```

1.2.13 Formatting Numbers

[19]:
```
# This program demonstrates how a floating-point
# number is displayed with no formatting.
amount_due = 5000.0
monthly_payment = amount_due / 12
print('The monthly payment is', monthly_payment)
```

```
The monthly payment is 416.6666666666667
```

[20]:
```
# This program demonstrates how a floating-point
# number can be formatted.
amount_due = 5000.0
monthly_payment = amount_due / 12
print('The monthly payment is', \
      format(monthly_payment, '.2f'))
```

```
The monthly payment is 416.67
```

1.2.14 Inserting Comma Separators

[21]:
```
# This program demonstrates how a floating-point
# number can be displayed as currency.
monthly_pay = 5000.0
annual_pay = monthly_pay * 12
print('Your annual pay is $', \
      format(annual_pay, ',.2f'), \
      sep='')
```

```
Your annual pay is $60,000.00
```

1.2.15 Specifying a Minimum Field Width

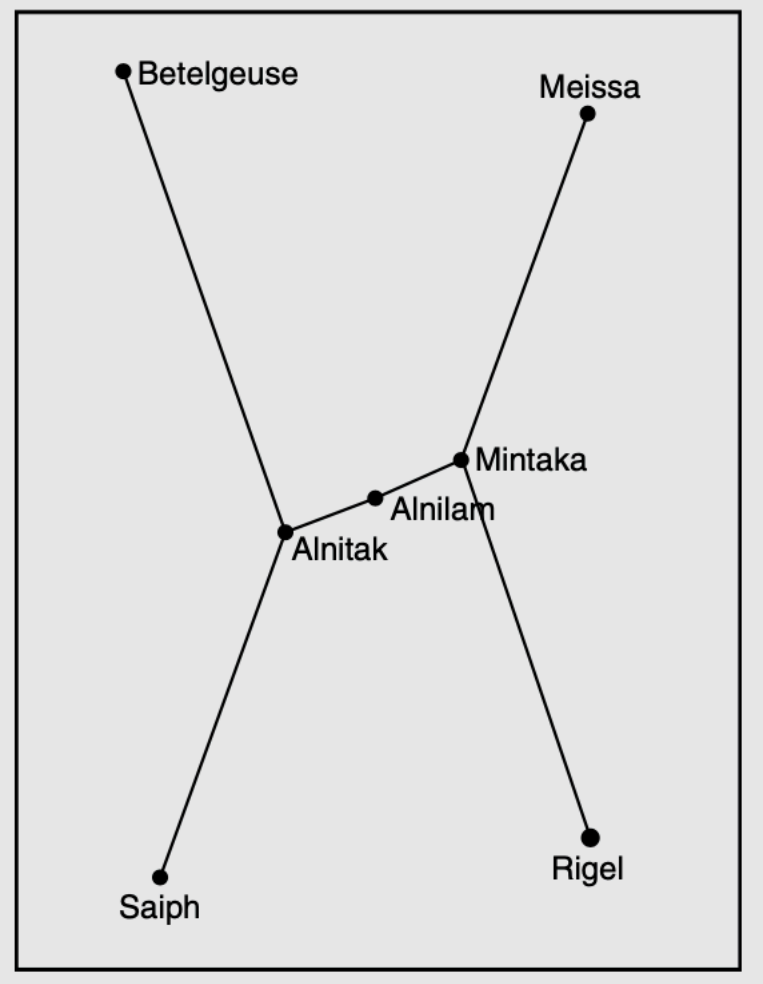
[22]:
```
# This program displays the following
# floating-point numbers in a column
# with their decimal points aligned.
num1 = 127.899
num2 = 3465.148
num3 = 3.776
num4 = 264.821
num5 = 88.081
num6 = 799.999

# Display each number in a field of 7 spaces
# with 2 decimal places.
```

```
print(format(num1, '7.2f'))
print(format(num2, '7.2f'))
print(format(num3, '7.2f'))
print(format(num4, '7.2f'))
print(format(num5, '7.2f'))
print(format(num6, '7.2f'))
```

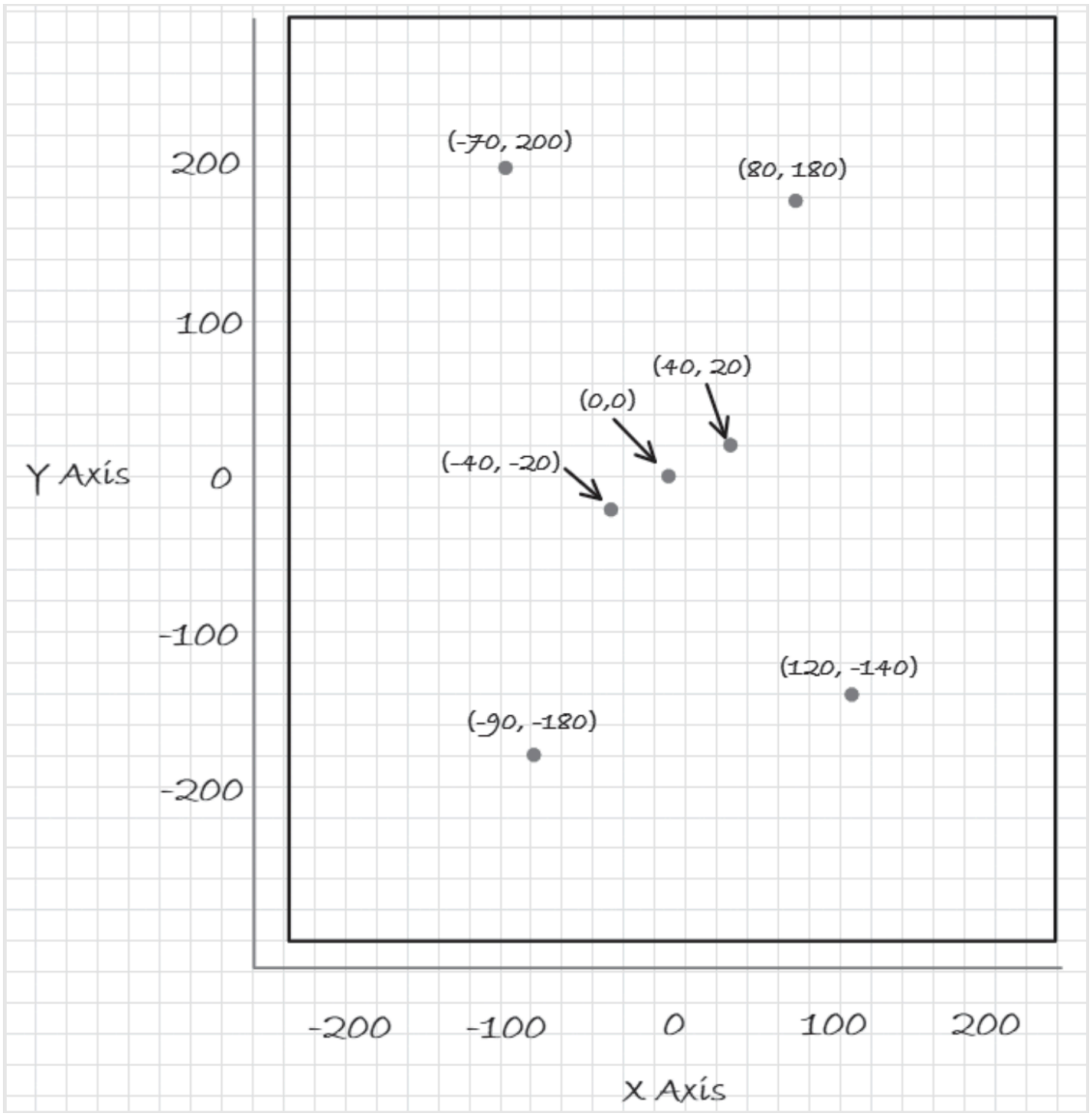
```
 127.90
3465.15
   3.78
 264.82
  88.08
 800.00
```

1.2.16 Turtle Graphics: The Orion Constellation Program

Orion is one of the most famous constellations in the night sky. The topmost stars are Orion's shoulders, the row of three stars in the middle are Orion's belt, and the bottom two stars are Orion's knees. The diagram in the below figure shows the names of each of these stars and the lines that are typically used to connect the stars.



In this section, we will develop a program that displays the stars, the star names, and the constellation lines as they are shown in the above figure. The program will display the constellation in a graphics window that is 500 pixels wide and 600 pixels high. The program will display dots to represent the stars. We will use a piece of graph paper, as shown in the following figure, to sketch the positions of the dots and determine their coordinates.



We will be using the coordinates that are identified in the above figure. As you can imagine, keeping track of the correct coordinates for each star can be difficult and tedious. To make things more simple in our code, we will create the following named constants to represent each star's coordinates:

LEFT_SHOULDER_X = −70 LEFT_SHOULDER_Y = 200

RIGHT_SHOULDER_X = 80 RIGHT_SHOULDER_Y = 180

LEFT_BELTSTAR_X = −40 LEFT_BELTSTAR_Y = −20

MIDDLE_BELTSTAR_X = 0 MIDDLE_BELTSTAR_Y = 0

RIGHT_BELTSTAR_X = 40 RIGHT_BELTSTAR_Y = 20

LEFT_KNEE_X = −90 LEFT_KNEE_Y = −180

RIGHT_KNEE_X = 120 RIGHT_KNEE_Y = −140

Now that we have identified the coordinates for the stars and created named constants to represent them, we can write pseudocode for the first part of the program, which displays the stars:

```
Set the graphics window size to 500 pixels wide by 600 pixels high
Draw a dot at (LEFT_SHOULDER_X,
               LEFT_SHOULDER_Y)                    # Left shoulder
Draw a dot at (RIGHT_SHOULDER_X,
               RIGHT_SHOULDER_Y)                   # Right shoulder
Draw a dot at (LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y)   # Leftmost star in the belt
Draw a dot at (MIDDLE_BELTSTAR_X,
               MIDDLE_BELTSTAR_Y)                  # Middle star in the belt
Draw a dot at (RIGHT_BELTSTAR_X,
               RIGHT_BELTSTAR_Y)                   # Rightmost star in the belt
Draw a dot at (LEFT_KNEE_X, LEFT_KNEE_Y)           # Left knee
Draw a dot at (RIGHT_KNEE_X, RIGHT_KNEE_Y)         # Right knee
```

Next, we will display the names of each star. The pseudocode for displaying these names follows.

```
Display the text "Betelgeuse" at (LEFT_SHOULDER_X,
                                  LEFT_SHOULDER_Y)     # Left shoulder
Display the text "Meissa" at (RIGHT_SHOULDER_X,
                              RIGHT_SHOULDER_Y)        # Right shoulder
Display the text "Alnitak" at (LEFT_BELTSTAR_X,
                               LEFT_ BELTSTAR_Y)       # Leftmost star in the belt
Display the text "Alnilam" at (MIDDLE_BELTSTAR_X,
                               MIDDLE_BELTSTAR_Y)      # Middle star in the belt
Display the text "Mintaka" at (RIGHT_BELTSTAR_X,
                               RIGHT_BELTSTAR_Y)       # Rightmost star in the belt
Display the text "Saiph" at (LEFT_KNEE_X,
                             LEFT_ KNEE_Y)             # Left knee
Display the text "Rigel" at (RIGHT_KNEE_X,
                             RIGHT_ KNEE_Y)            # Right knee
```

Next, we will display the lines that connect the stars. The pseudocode for displaying these lines follows.

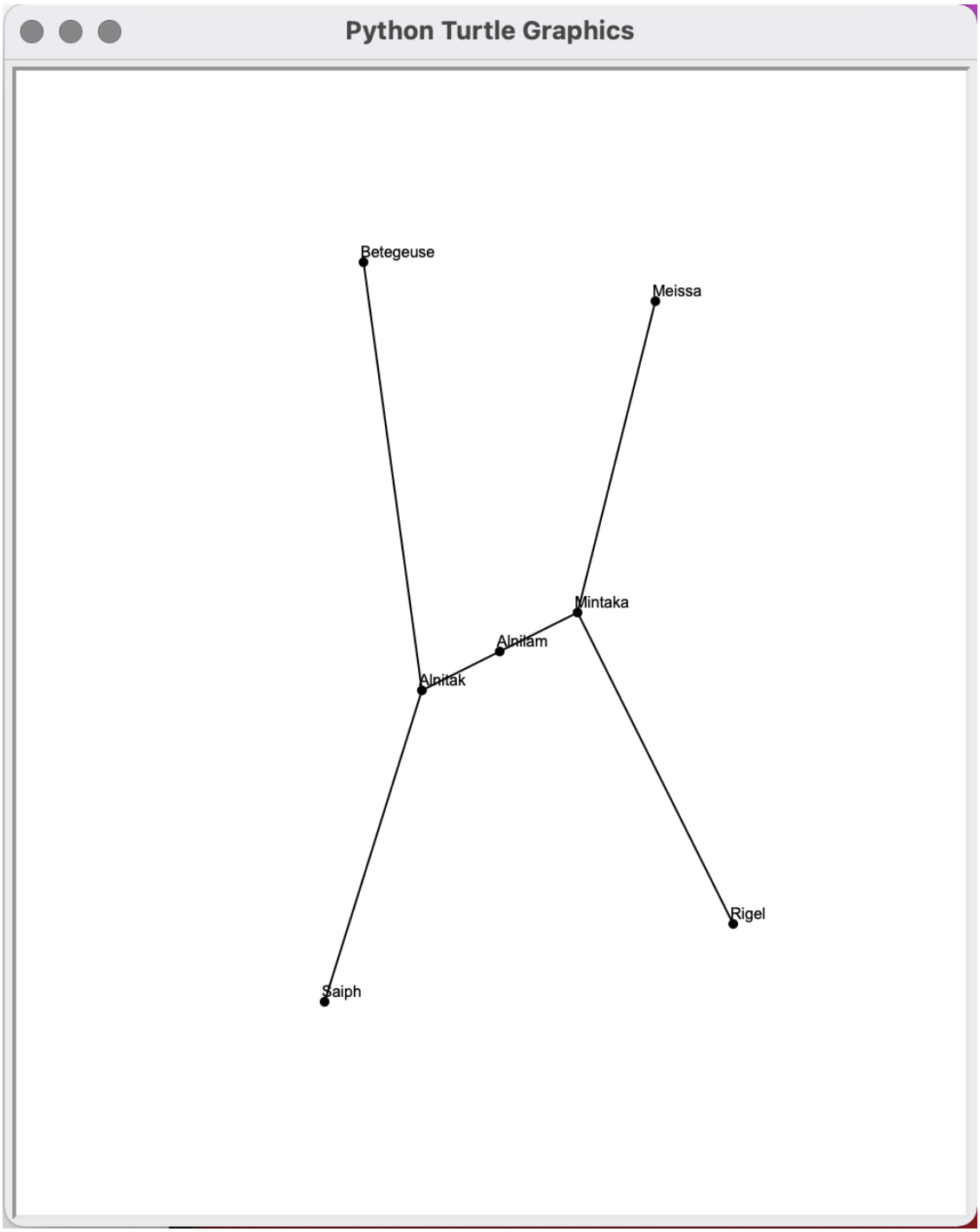
```
# Left shoulder to left belt star
Draw a line from (LEFT_SHOULDER_X, LEFT_SHOULDER_Y) to
(LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y)

# Right shoulder to right belt star
Draw a line from (RIGHT_SHOULDER_X, RIGHT_SHOULDER_Y) to
(RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y)

# Left belt star to middle belt star
Draw a line from (LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y) to
(MIDDLE_BELTSTAR_X, MIDDLE_BELTSTAR_Y)

# Middle belt star to right belt star
Draw a line from (MIDDLE_BELTSTAR_X, MIDDLE_BELTSTAR_Y) to
(RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y)

# Left belt star to left knee
Draw a line from (LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y) to
(LEFT_KNEE_X, LEFT_KNEE_Y)

# Right belt star to right knee
Draw a line from (RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y) to
(RIGHT_KNEE_X, RIGHT_KNEE_Y)
```

Now that we know the logical steps that the program must perform, we are ready to start writing code. When the program runs, it first displays the stars, then it displays the names of the stars, and then it displays the constellation lines.

[23]:
```
# This program draws the stars of the Orion constellation,
# the names of the stars, and the constellation lines.
import turtle

# Set the window size.
turtle.setup(500, 600)

# Setup the turtle.
turtle.penup()
turtle.hideturtle()

# Create named constants for the star coordinates.
LEFT_SHOULDER_X = -70
LEFT_SHOULDER_Y = 200

RIGHT_SHOULDER_X = 80
RIGHT_SHOULDER_Y = 180

LEFT_BELTSTAR_X = -40
LEFT_BELTSTAR_Y = -20

MIDDLE_BELTSTAR_X = 0
```

```
MIDDLE_BELTSTAR_Y = 0

RIGHT_BELTSTAR_X = 40
RIGHT_BELTSTAR_Y = 20

LEFT_KNEE_X = -90
LEFT_KNEE_Y = -180

RIGHT_KNEE_X = 120
RIGHT_KNEE_Y = -140

# Draw the stars.
turtle.goto(LEFT_SHOULDER_X, LEFT_SHOULDER_Y)     # Left shoulder
turtle.dot()
turtle.goto(RIGHT_SHOULDER_X, RIGHT_SHOULDER_Y)   # Right shoulder
turtle.dot()
turtle.goto(LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y)     # Left belt star
turtle.dot()
turtle.goto(MIDDLE_BELTSTAR_X, MIDDLE_BELTSTAR_Y) # Middle belt star
turtle.dot()
turtle.goto(RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y)   # Right belt star
turtle.dot()
turtle.goto(LEFT_KNEE_X, LEFT_KNEE_Y)             # Left knee
turtle.dot()
turtle.goto(RIGHT_KNEE_X, RIGHT_KNEE_Y)           # Right knee
turtle.dot()

# Display the star names
turtle.goto(LEFT_SHOULDER_X, LEFT_SHOULDER_Y)     # Left shoulder
turtle.write('Betegeuse')
turtle.goto(RIGHT_SHOULDER_X, RIGHT_SHOULDER_Y)   # Right shoulder
turtle.write('Meissa')
turtle.goto(LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y)     # Left belt star
turtle.write('Alnitak')
turtle.goto(MIDDLE_BELTSTAR_X, MIDDLE_BELTSTAR_Y) # Middle belt star
turtle.write('Alnilam')
turtle.goto(RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y)   # Right belt star
turtle.write('Mintaka')
turtle.goto(LEFT_KNEE_X, LEFT_KNEE_Y)             # Left knee
turtle.write('Saiph')
turtle.goto(RIGHT_KNEE_X, RIGHT_KNEE_Y)           # Right knee
turtle.write('Rigel')

# Draw a line from the left shoulder to left belt star
turtle.goto(LEFT_SHOULDER_X, LEFT_SHOULDER_Y)
turtle.pendown()
turtle.goto(LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y)
```

```
turtle.penup()

# Draw a line from the right shoulder to right belt star
turtle.goto(RIGHT_SHOULDER_X, RIGHT_SHOULDER_Y)
turtle.pendown()
turtle.goto(RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y)
turtle.penup()

# Draw a line from the left belt star to middle belt star
turtle.goto(LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y)
turtle.pendown()
turtle.goto(MIDDLE_BELTSTAR_X, MIDDLE_BELTSTAR_Y)
turtle.penup()

# Draw a line from the middle belt star to right belt star
turtle.goto(MIDDLE_BELTSTAR_X, MIDDLE_BELTSTAR_Y)
turtle.pendown()
turtle.goto(RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y)
turtle.penup()

# Draw a line from the left belt star to left knee
turtle.goto(LEFT_BELTSTAR_X, LEFT_BELTSTAR_Y)
turtle.pendown()
turtle.goto(LEFT_KNEE_X, LEFT_KNEE_Y)
turtle.penup()

# Draw a line from the right belt star to right knee
turtle.goto(RIGHT_BELTSTAR_X, RIGHT_BELTSTAR_Y)
turtle.pendown()
turtle.goto(RIGHT_KNEE_X, RIGHT_KNEE_Y)

# Keep the window open. (Not necessary with IDLE.)
turtle.done()
```



1.2.17 Reference

Aforementioned contents are adapted from the following book: * Gaddis, T. Starting out with Python, 4th Ed., Pearson Education, 2018.