Blain's World @ ACC — cti110 class

version 6.0

Last generated: October 29, 2021

Table of Contents

CTI-110

MindTap Tips

Web

Programing

Chapter 1

Project 1.1 3
Project 1.2 4
Project 1.3 5
Project 1.4 6
Project 1.5 7
Project 1.6 8
Project 1.7 9
Project 1.8 10
Project 1.9 11
Project 1.10 12

Chapter 2

Project 2.1 13
Project 2.2 14
Project 2.3 15
Project 2.4 16
Project 2.5 18
Project 2.6 19
Project 2.7 20
Project 2.8 21
Project 2.9 21
Project 2.10 23

Chapter 3

Project 3.1 24
Project 3.2 25
Project 3.3 27

Project 3.4 30
Project 3.5 32
Project 3.6 34
Project 3.7 36
Project 3.8 38
Project 3.9 38
Project 3.10 42
Project 3.11 46

Chapter 4

Project 4.1 48
Project 4.2 50
Project 4.3 52
Project 4.4 55
Project 4.5 56
Project 4.6 58
Project 4.7 60
Project 4.8 61
Project 4.9 62
Project 4.10 63
Project 4.11 64
Project 4.12 69

Database

Project 1.1

Summary: New content coming soon

under construction

Project 1.2

Summary: New content coming soon

under construction

Project 1.3

Summary: New content coming soon

under construction

Project 1.4

Summary: New content coming soon

under construction

Project 1.5

Summary: New content coming soon

under construction

Project 1.6

Summary: New content coming soon

under construction

Project 1.7

Summary: New content coming soon

under construction

Project 1.8

Summary: New content coming soon

under construction

Project 1.9

Summary: New content coming soon

under construction

Project 1.10

Summary: New content coming soon

under construction

Project 2.1

Summary: New content coming soon

under construction

Project 2.2

Summary: New content coming soon

under construction

Project 2.3

Summary: New content coming soon

under construction

Project 2.4

Instructions

Write a program that takes the radius of a sphere (a floating-point number) as input and then outputs the sphere's:

1. Diameter (2 × radius)
2. Circumference (diameter × π)
3. Surface area (4 × π × radius2)
4. Volume (4/3 × π × radius3)

For convenience, the program can import the math module.

Below is an example of the program input and output:

```
Radius = 5

Diameter     : 10.0
Circumference: 31.41592653589793
Surface area : 314.1592653589793
Volume       : 523.5987755982989
```

Flowchart

Starter Code

```
"""
Program: sphere.py
Project 2.4

Given the radius compute the diameter, circumference, and volum
e
of a sphere.

Useful facts:
   diameter = 2 * radius
   circumference = diameter * PI
   surface area = 4 * PI * radius * radius
   volume = 4/3 * PI * radius * radius * radius

"""
# import modules

# Request the input

# Compute the results

# Display the results
```

Project 2.5

Summary: New content coming soon

under construction

Project 2.6

Summary: New content coming soon

under construction

Project 2.7

Summary: New content coming soon

under construction

Project 2.8

Summary: New content coming soon

under construction

Project 2.8

Summary: New content coming soon

under construction

Project 2.10

Summary: New content coming soon

under construction

Project 3.1

Instructions

Write a program that accepts the lengths of three sides of a triangle as inputs.

The program output should indicate whether or not the triangle is an equilateral triangle.

Use The triangle is equilateral. and The triangle is not equilateral. as your final outputs. An example of the program inputs and output is shown below:

```
Enter the first side: 2
Enter the second side: 2
Enter the third side: 2

The triangle is equilateral.
```

Flowchart

equilateral flowchart

Starter Code

```
"""
Program: equilateral.py
Project 3.1

Determine whether or not three input sides compose an
equilateral triangle.
"""

# Request the inputs


# Determine the result and display it
```

Project 3.2

Instructions

Write a program that accepts the lengths of three sides of a triangle as inputs. The program output should indicate whether or not the triangle is a right triangle.

Recall from the Pythagorean theorem that in a right triangle, the square of one side equals the sum of the squares of the other two sides.

Use The triangle is a right triangle. and The triangle is not a right triangle. as your final outputs. An example of the program input and proper output format is shown below:

```
Enter the first side : 3
Enter the second side: 4
Enter the third side : 5

The triangle is a right triangle.
```

> Any of the three sides of the triangle could be the hypotenuse. The program should handle this appropriately.

Flowchart

right flowchart

Starter Code

```
"""
Program: right.py
Project 3.2

Determine whether or not three input sides compose a
right triangle.
"""

# Request the inputs

# Compute the squares

# Determine the result and display it
```

Project 3.3

Instructions

Modify the guessing-game program so that the user thinks of a number that the computer must guess.

- The computer must make no more than the minimum number of guesses, and it must prevent the user from cheating by entering misleading hints.
- Use **I'm out of guesses, and you cheated** and **Hooray, I've got it in X tries** as your final output.

> (Hint: Use the math.log function to compute the minimum number of guesses needed after the lower and upper bounds are entered.)

Below are two test runs of the program:

```
Enter the smaller number: 0
Enter the larger number: 10

0 10
Your number is 5
Enter =, <, or >: <
0 4
Your number is 2
Enter =, <, or >: >
3 4
Your number is 3
Enter =, <, or >: =
Hooray, I've got it in 3 tries!
Enter the smaller number: 0
Enter the larger number: 50
0 50
Your number is 25
Enter =, <, or >: <
0 24
Your number is 12
Enter =, <, or >: <
0 11
Your number is 5
Enter =, <, or >: <
0 4
Your number is 2
Enter =, <, or >: <
0 1
Your number is 0
Enter =, <, or >: >
1 1
Your number is 1
Enter =, <, or >: >
I'm out of guesses, and you cheated!
```

Flowchart

guess flowchart

Starter Code

```
"""
Program: guess.py
Project 3.3

The computer guesses the user's number using the minimum
number of attempts and prevents cheating by the user.
"""

import random

smaller = int(input("Enter the smaller number: "))
larger = int(input("Enter the larger number: "))
myNumber = random.randint(smaller, larger)
count = 0
while True:
    count += 1
    userNumber = int(input("Enter your guess: "))
    if userNumber < myNumber:
        print("Too small")
    elif userNumber > myNumber:
        print("Too large")
    else:
        print("You've got it in", count, "tries!")
        break
```

Project 3.4

Instructions

A standard science experiment is to drop a ball and see how high it bounces. Once the “bounciness” of the ball has been determined, the ratio gives a bounciness index.

For example, if a ball dropped from a height of 10 feet bounces 6 feet high, the index is 0.6, and the total distance traveled by the ball is 16 feet after one bounce. If the ball were to continue bouncing, the distance after two bounces would be 10 ft + 6 ft +6 ft + 3.6 ft = 25.6 ft. Note that the distance traveled for each successive bounce is the distance to the floor plus 0.6 of that distance as the ball comes back up.

Write a program that lets the user enter the initial height from which the ball is dropped, the bounciness index, and the number of times the ball is allowed to continue bouncing. Output should be the total distance traveled by the ball.

Below is an example of the program input and output:

```
Enter the height from which the ball is dropped: 25
Enter the bounciness index of the ball: .5
Enter the number of times the ball is allowed to continue bounc
ing: 3

Total distance traveled is: 65.625 units.
```

Flowchart

bounce flowchart

Starter Code

```
"""
Program: bouncy.py
Project 3.4

This program calculates the total distance a ball travels as i
t bounces given:
1. the initial height of the ball
2. its bounciness index
3. the number of times the ball is allowed to continue bouncing

"""
# Request the inputs

# Initialize loop variables

# Loop through till out of bounces

    # Calculate the distance traveled

# Output distance traveled
```

Project 3.5

Instructions

A local biologist needs a program to predict population growth. The inputs would be:

1. The initial number of organisms, as an **int**
2. The rate of growth (a real number greater than 1), as a **float**
3. The number of hours it takes to achieve this rate, as an **int**
4. A number of hours during which the population grows, as an **int**

For example, one might start with a population of 500 organisms, a growth rate of 2, and a growth period to achieve this rate of 6 hours. Assuming that none of the organisms die, this would imply that this population would double in size every 6 hours. Thus, after allowing 6 hours for growth, we would have 1000 organisms, and after 12 hours, we would have 2000 organisms.

Write a program that takes these inputs and displays a prediction of the total population.

An example of the program input and output is shown below:

```
Enter the initial number of organisms: 10
Enter the rate of growth [a real number > 1]: 2
Enter the number of hours to achieve the rate of growth: 2
Enter the total hours of growth: 6

The total population is 80
```

Flowchart

Population flowchart

Starter Code

```
"""
Program: population.py
Project 3.5

Predict population growth, assuming that no
organisms die.

Inputs:
   initial number of organisms
   rate of growth (a float > 1)
   the number of hours to achieve the rate
   number of hours of growth
"""

# Accept the inputs

# Calculate the number of cycles

# Calculate the population after an integral number of cycles

# Output the total population
```

Project 3.6

Instructions

The German mathematician Gottfried Leibniz developed the following method to approximate the value of π:

π/4 = 1 - 1/3 + 1/5 - 1/7 + . . .

Write a program that allows the user to specify the number of iterations used in this approximation and that displays the resulting value.

An example of the program input and output is shown below:

```
Enter the number of iterations: 5

The approximation of pi is 3.3396825396825403
```

Flowchart

---

flowchart

Starter Code

```
"""
Program: leibniz.py
Project 3.6
This program approximates the value of pi using an algorithm
designed by the German mathematician Gottfried Leibniz. The
algorithm is as follows:
pi = 4 - 4 / 3 + 4 / 5 - 4 / 7 + . . .
This program allows the user to specify the number of iteration
s
to use in the approximation.
"""
# Request the inputs

# Initialize loop variables

# Loop through iterations

    # Calculate π/4 for this iteration

# Output results
```

Project 3.7

Instructions

Teachers in most school districts are paid on a schedule that provides a salary based on their number of years of teaching experience.

For example, a beginning teacher in the Lexington School District might be paid $30,000 the first year. For each year of experience after this first year, up to 10 years, the teacher receives a 2% increase over the preceding value.

Write a program that displays a salary schedule, in tabular format, for teachers in a school district. The inputs are:

1. Starting salary
2. Annual percentage increase
3. Number of years for which to print the schedule

Each row in the schedule should contain the year number and the salary for that year

An example of the program input and output is shown below:

```
Enter the starting salary: $30000
Enter the annual % increase: 2
Enter the number of years: 10

Year   Salary
-------------
 1    30000.00
 2    30600.00
 3    31212.00
 4    31836.24
 5    32472.96
 6    33122.42
 7    33784.87
 8    34460.57
 9    35149.78
10    35852.78
```

Flowchart

flowchart

Starter Code

```
"""
Program: salary.py
Project 3.7

Compute a school district's salary schedule.

Inputs
   starting salary
   annual percentage increase
   number of years for which to print the schedule

Outputs
    Two columns containing the year and the salary
    after the increase.
"""

# Accept the inputs

# Output Header

# Compute and display the results

    # calculate next salary
```

Project 3.8

Instructions

The greatest common divisor of two positive integers, A and B, is the largest number that can be evenly divided into both of them. Euclid's algorithm can be used to find the greatest common divisor (GCD) of two positive integers. You can use this algorithm in the following manner:

1. Compute the remainder of dividing the larger number by the smaller number.
2. Replace the larger number with the smaller number and the smaller number with the remainder.
3. Repeat this process until the smaller number is zero.

The larger number at this point is the GCD of A and B. Write a program that lets the user enter two integers and then prints each step in the process of using the Euclidean algorithm to find their GCD.

An example of the program input and output is shown below:

```
Enter the smaller number: 5
Enter the larger number: 15

The greatest common divisor is 5
```

Flowchart

---

gcd flowchart

Starter Code

```
"""
Program: gcd.py
Project 3.8

Compute and print the greatest common divisor of two input
integers.
"""
# Accept the inputs

# Repeat this process until the smaller number is zero

    # Compute the remainder of dividing the larger number by th
e smaller number.

    # Replace the larger number with the smaller number and th
e smaller number with the remainder.

# Output the greatest common divisor
```

Project 3.8

Instructions

The greatest common divisor of two positive integers, A and B, is the largest number that can be evenly divided into both of them. Euclid's algorithm can be used to find the greatest common divisor (GCD) of two positive integers. You can use this algorithm in the following manner:

1. Compute the remainder of dividing the larger number by the smaller number.
2. Replace the larger number with the smaller number and the smaller number with the remainder.
3. Repeat this process until the smaller number is zero.

The larger number at this point is the GCD of A and B. Write a program that lets the user enter two integers and then prints each step in the process of using the Euclidean algorithm to find their GCD.

An example of the program input and output is shown below:

```
Enter the smaller number: 5
Enter the larger number: 15

The greatest common divisor is 5
```

Flowchart

---

gcd flowchart

Starter Code

```
"""
Program: gcd.py
Project 3.8

Compute and print the greatest common divisor of two input
integers.
"""
# Accept the inputs

# Repeat this process until the smaller number is zero

    # Compute the remainder of dividing the larger number by th
e smaller number.

    # Replace the larger number with the smaller number and th
e smaller number with the remainder.

# Output the greatest common divisor
```

Project 3.10

The credit plan at TidBit Computer Store specifies a 10% down payment and an annual interest rate of 12%. Monthly payments are 5% of the listed purchase price, minus the down payment.

Write a program that takes the purchase price as input. The program should display a table, with appropriate headers, of a payment schedule for the lifetime of the loan. Each row of the table should contain the following items:

1. The month number (beginning with 1)
2. The current total balance owed
3. The interest owed for that month
4. The amount of principal owed for that month
5. The payment for that month
6. The balance remaining after payment
7. The amount of interest for a month is equal to balance × rate / 12.

The amount of principal for a month is equal to the monthly payment minus the interest owed.

Example Input Output

An example of the program input and output is shown below:

```
Enter the purchase price: 200

Month  Starting Balance  Interest to Pay  Principal to Pay  Pay
ment  Ending Balance
 1         180.00           1.80             7.20
9.00           172.80
 2         172.80           1.73             7.27
9.00           165.53
 3         165.53           1.66             7.34
9.00           158.18
 4         158.18           1.58             7.42
9.00           150.77
 5         150.77           1.51             7.49
9.00           143.27
 6         143.27           1.43             7.57
9.00           135.71
 7         135.71           1.36             7.64
9.00           128.06
 8         128.06           1.28             7.72
9.00           120.34
 9         120.34           1.20             7.80
9.00           112.55
10         112.55           1.13             7.87
9.00           104.67
11         104.67           1.05             7.95
9.00            96.72
12          96.72           0.97             8.03
9.00            88.69
13          88.69           0.89             8.11
9.00            80.57
14          80.57           0.81             8.19
9.00            72.38
15          72.38           0.72             8.28
9.00            64.10
16          64.10           0.64             8.36
9.00            55.74
17          55.74           0.56             8.44
9.00            47.30
18          47.30           0.47             8.53
9.00            38.77
19          38.77           0.39             8.61
9.00            30.16
20          30.16           0.30             8.70
9.00            21.46
21          21.46           0.21             8.79
```

```
9.00            12.68
22          12.68           0.13            8.87
9.00             3.80
23           3.80           0.00            3.80
3.80             0.00
```

Flowchart

tidbit flowchart

Starter Code

```
"""
Program: tidbit.py
Project 1.5

Print a payment schedule for a loan to purchase a computer.

Input
   purchase price

Constants
   annual interest rate = 12%
   downpayment = 10% of purchase price
   monthly payment = 5% of purchase price

"""
# Set Constants


# Request input


# Calculate down payment and purchase price


# Calculate monthly payment


# Output schedule header


# Initialize loop variables

# Loop through declining balance

    # Output "Month  Starting Balance  Interest to Pay  Princip
al to Pay  Payment  Ending Balance" on this iteration

    # Iterate loop variables
```

Project 3.11

Instructions

In the game of Lucky Sevens, the player rolls a pair of dice. If the dots add up to 7, the player wins $4; otherwise, the player loses $1.

Suppose that, to entice the gullible, a casino tells players that there are lots of ways to win: (1, 6), (2, 5), and so on. A little mathematical analysis reveals that there are not enough ways to win to make the game worthwhile; however, because many people's eyes glaze over at the first mention of mathematics, your challenge is to write a program that demonstrates the futility of playing the game.

Your program should take as input the amount of money that the player wants to put into the pot, and using a random number generator play the game until the pot is empty. At that point, the program should print:

1. The number of rolls it took to break the player
2. The maximum amount of money in the pot.

An example of the program input and output is shown below:

```
How many dollars do you have? 50

You are broke after 220 rolls.
You should have quit after 6 rolls when you had $59.
```

> This program implements logic to create semi-random results. The example above is one in many possible scenarios.

Flowchart

sevens flowchart

Starter Code

```
"""
Program: sevens.py
Project 3.11

Simulate the game of lucky sevens until all funds are depleted.

1) Rules:
       roll two dice
       if the sum equals 7, win $4, else lose $1
2) The input is:
       the amount of money the user is prepared to lose
3) Computations:
       use a random number generator to simulate rolling the di
ce
       loop until the funds are depleted
       count the number of rolls
       keep track of the maximum amount
4) The outputs are:
       the number of rolls it takes to deplete the funds
       the maximum amount

"""
# import module

# Request the input

# Initialize variables

# Loop until the money is gone

    # Roll the dice

    #Calculate the winnings or losses

    #If this is a new maximum, remember it


# Display the results
```

Project 4.1

Instructions

Write a script that inputs a line of plaintext and a distance value and outputs an encrypted text using a Caesar cipher.

The script should work for any printable characters.

An example of the program input and output is shown below:

```
Enter a message: Hello world!
Enter the distance value: 4

Lipps${svph%
```

Flowchart

encrypt flowchart

Starter Code

```
"""
File: encrypt.py
Project 4.1

Encrypts an input string of the ASCII characters and prints
the result.  The other input is the distance value.
"""

# The ASCII values range from 0 through 127
# Hint: ord('a') = 97 and ord('z') = 122
```

Image from textbook section 4-2 Data Encryption

Image from Textbook

Project 4.2

Instructions

Write a script that inputs a line of encrypted text and a distance value and outputs plaintext using a Caesar cipher.

The script should work for any printable characters.

An example of the program input and output is shown below:

```
Enter the coded text: Lipps${svph%
Enter the distance value: 4

Hello world!
```

Flowchart

decrypt flowchart

Starter Code

```
"""
File: decrypt.py
Project 4.2

Decrypts an input string characters and prints
the result.  The other input is the distance value.
"""

# The ASCII values range from 0 through 127
# Hint: ord('a') = 97 and ord('z') = 122
```

Image from textbook section 4-2 Data Encryption

Image from Textbook

Project 4.3

Instructions

Modify the scripts of Projects 1 and 2 to encrypt and decrypt entire files of text. An example of the program interface is shown below:

```
Enter the input file name: encrypted.txt
Enter the output file name: a
Enter the distance value: 3
```

Flowchart

Encrypt

encrypt flowchart

Decrypt

decrypt flowchart

Starter Code

```
"""
File: encrypt.py
Probject 4.1

Encypts an input string of the ASCII characters and prints
the result.  The other input is the distance value.
"""

# The ASCII values range from 0 through 127

plainText = input("Enter a message: ")
distance = int(input("Enter the distance value: "))
code = ""
for ch in plainText:
    ordValue = ord(ch)
    cipherValue = ordValue + distance
    if cipherValue > 127:
        cipherValue = distance - (127 - ordValue + 1)
    code +=  chr(cipherValue)
print(code)
```

```
"""
File: decrypt.py
Project 4.2

Decrypts an input string characters and prints
the result.  The other input is the distance value.
"""

# The ASCII values range from 0 through 127

code = input("Enter the coded text: ")
distance = int(input("Enter the distance value: "))
plainText = ''
for ch in code:
    ordValue = ord(ch)
    cipherValue = ordValue - distance
    if cipherValue < 0:
        cipherValue = 127 - \
                      (distance - (1 - ordValue))
    plainText +=  chr(cipherValue)
print(plainText)
```

Project 4.4

Instructions

Octal numbers have a base of eight and the digits 0–7. Write the scripts octalToDecimal.py and decimalToOctal.py, which convert numbers between the octal and decimal representations of integers.

These scripts use algorithms that are similar to those of the binaryToDecimal and decimalToBinary scripts developed in the Section: Strings and Number Systems.

An example of octalToDecimal.py input and output is shown below:

```
Enter a string of octal digits: 234

The integer value is 156
```

An example of decimalToOctal.py input and output is shown below:

```
Enter a decimal integer: 27

Quotient Remainder Octal
    3       3           3
    0       3          33
The octal representation is 33
```

FlowChart

Decimal To Octal

dec to oct

Octal To Decimal

oct to dec

Project 4.5

Instructions

A bit shift is a procedure whereby the bits in a bit string are moved to the left or to the right.

For example, we can shift the bits in the string 1011 two places to the left to produce the string 1110. Note that the leftmost two bits are wrapped around to the right side of the string in this operation.

Define two scripts, shiftLeft.py and shiftRight.py, that expect a bit string as an input.

The script shiftLeft shifts the bits in its input one place to the left, wrapping the leftmost bit to the rightmost position. The script shiftRight performs the inverse operation.

Each script prints the resulting string.

An example of shiftLeft.py input and output is shown below:

```
Enter a string of bits: Hello world!

ello world!H
```

An example of shiftRight.py input and output is shown below:

```
Enter a string of bits: Hello world!

!Hello world
```

FlowChart

Shift Left

Shift Left

Shift Right

Shift Left

Starter Code

```
"""
File: shiftleft.py
Project 4.5

Shifts the bits in an input string one place to the left.
The leftmost bit wraps around to the rightmost position.
"""
```

```
"""
File: shiftright.py
Project 4.5

Shifts the bits in an input string one place to the right.
The rightmost bit wraps around to the leftmost position.
"""
```

Project 4.6

Instructions

Use the strategy of the decimal to binary conversion implemented in Project 4.4, and the bit shift left operation defined in Project 4.5 to code a new encryption algorithm.

The algorithm should

- Add 1 to each character's numeric ASCII value.
- Convert it to a bit string.
- Shift the bits of this string one place to the left.

A single-space character in the encrypted string separates the resulting bit strings.

An example of the program input and output is shown below:

```
Enter a message: Hello world!

0010011 1001101 1011011 1011011 1100001 000011 1110001 1100001
1100111 1011011 1001011 000101
```

Flowchart

encrypt flowchart

Starter Code

```
"""
File: encrypt.py
Probject 4.6

Encypts an input string of characters and prints
the result.
"""

# Prompt user to enter a message

# Initialize variable to store encrypted text

# iterate

    # Add 1 to ASCII value

    # Convert to binary



    # Shift one bit to left

    # Add encrypted character to code string
```

Project 4.7

Summary: New content coming soon

under construction

”—\ntitle: Project 2.”$x”\ntags: [programming]\nkeywords: notes, tips, cautions, warnings, admonitions\nlast_updated: October 23, 2021\nsummary: “New content coming soon”\nsidebar: cti110_sidebar\npermalink: prog_2”$x”.7_Readme.html\nfolder: cti110\n—\n\n$_{\text{under construction}}$\n\n

\n”

Project 4.8

Summary: New content coming soon

under construction

Project 4.9

Summary: New content coming soon

under construction

Project 4.10

Summary: New content coming soon

under construction

Project 4.11

Summary: You can insert notes, tips, warnings, and important alerts in your content. These notes are stored as shortcodes made available through the linksrefs.hmtl include.

Instructions

Jack just completed the program for the Flesch text analysis from this chapter's case study. His supervisor, Jill, has discovered an error in his code. The error causes the program to count a syllable containing consecutive vowels as multiple syllables.

Suggest a solution to this problem in Jack's code and modify the program so that it handles these cases correctly.

An example text and the program input and output is shown below:

example.txt

```
Or to take arms against a sea of troubles, And by opposing end
them? To die: to sleep.
```

```
Enter the file name: example.txt
The Flesch Index is 102.045
The Grade Level Equivalent is 1
3 sentences
18 words
21 syllables
```

Flowcharts

Start file flowchart

textanalysis flowchart

Final Flowchart

textanalysis flowchart

Starter Code

```
"""
Program: textanalysis.py
Author: Ken
Computes and displays the Flesch Index and the Grade
Level Equivalent for the readability of a text file.
"""

# Take the inputs
fileName = input("Enter the file name: ")
inputFile = open(fileName, 'r')
text = inputFile.read()

# Count the sentences
sentences = text.count('.') + text.count('?') + \
    text.count(':') + text.count(';') + \
        text.count('!')

# Count the words
words = len(text.split())

# Count the syllables
syllables = 0
vowels = "aeiouAEIOU"
for word in text.split():
    for vowel in vowels:
        syllables += word.count(vowel)
    for ending in ['es', 'ed', 'e']:
        if word.endswith(ending):
            syllables -= 1
    if word.endswith('le'):
        syllables += 1

# Compute the Flesch Index and Grade Level
index = 206.835 - 1.015 * (words / sentences) - \
    84.6 * (syllables / words)
level = int(round(0.39 * (words / sentences) + 11.8 *
                  (syllables / words) - 15.59))

# Output the results
print("The Flesch Index is", index)
print("The Grade Level Equivalent is", level)
print(sentences, "sentences")
print(words, "words")
print(syllables, "syllables")
```

Test File

example.txt (page 0)

Project 4.12

Instructions

The Payroll Department keeps a list of employee information for each pay period in a text file. The format of each line of the file is the following: **<last name> <hours worked> <hourly wage>**

Write a program that inputs a filename from the user and prints to the terminal a report of the wages paid to the employees for the given period.

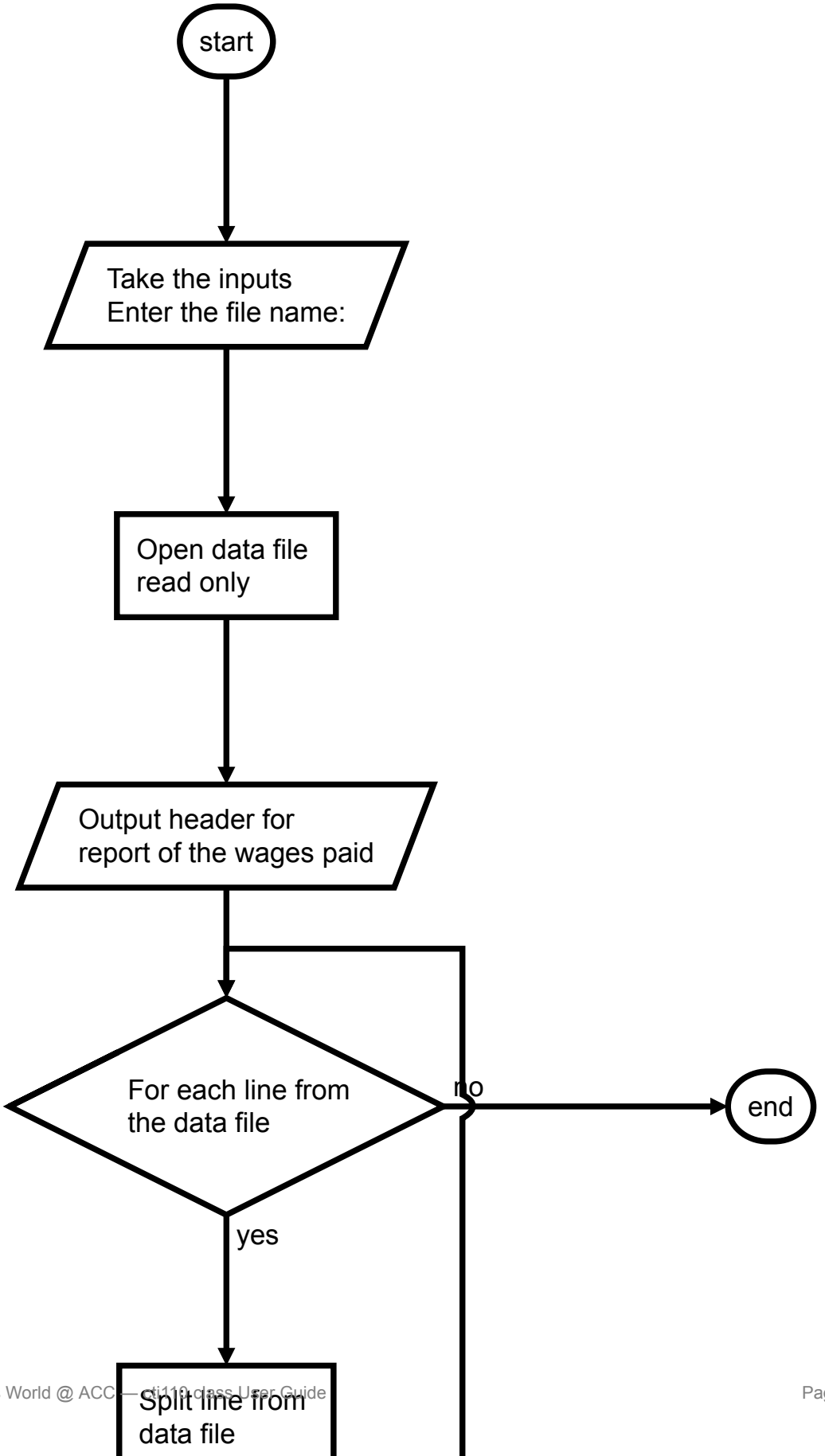
- The report should be in tabular format with the appropriate header.
- Each line should contain:
    - An employee's name
    - The hours worked
    - The wages paid for that period.

An example of the program input and output is shown below:

```
Enter the file name: data.txt

Name            Hours      Total Pay
Lambert            34         357.00
Osborne            22         137.50
Giacometti          5         503.50
```

Flowchart


start
Take the inputs
Enter the file name:
Open data file
read only
Output header for
report of the wages paid
For each line from
the data file
no
end
yes
Split line from
data file

Starter Code

```
"""
Program: payroll.py
Project 4.12

Print a payroll report.

Input
   A file in which each line has the form

   <last name> <hourly wage> <hours worked>

Output
   A report in tabular format.  Each line has the form

   <last name> <hours worked> <total wages>

"""

# Take the inputs


# Open the data file read only


# Read the data and print the report
# Output header for report of the wages paid

# For each line from the data file

    # Split line from data file into a list

    # Assign data from list to variables with appropriate datat
ype

    # Calculate Total Pay

    # Output Name, Hours, & Total Pay for this line from the da
ta file
```

Data File

```
Lambert 34 10.50
Osborne 22 6.25
Giacometti 5 100.70
```