ENDG 233 – Programming with Data

Algorithms - Flowchart, Pseudocode

UNIVERSITY OF CALGARY

Week 2: Sept. 13th – Sept. 19th

Learning Outcome

1. Solve algorithmic logic problems using flowcharts and pseudocode.

This week in glance

- Algorithmic thinking
  - Pseudocode
  - Flowchart
  - Examples
  - Zylab examples
- Python installation and run a simple program in Virtual Environment
- In-lab Exercise with Zylab (activated at last 75 minutes of the last session in this week)

Algorithm Thinking

- WHAT are you going to build?
- WHAT is the program's input?
- WHAT is the program's output?

Algorithm Thinking

- Algorithms should be
  - **Unambiguous** - each step has precise instructions, and it is clear where to go next.
  - **Executable** - must be possible for each step to be carried out in practice.
  - **Terminating** - the steps must eventually come to an end.

Omelette Algorithm (recipe)

1. In a separate bowl, combine two eggs
2. ....

Omelette Algorithm (recipe)

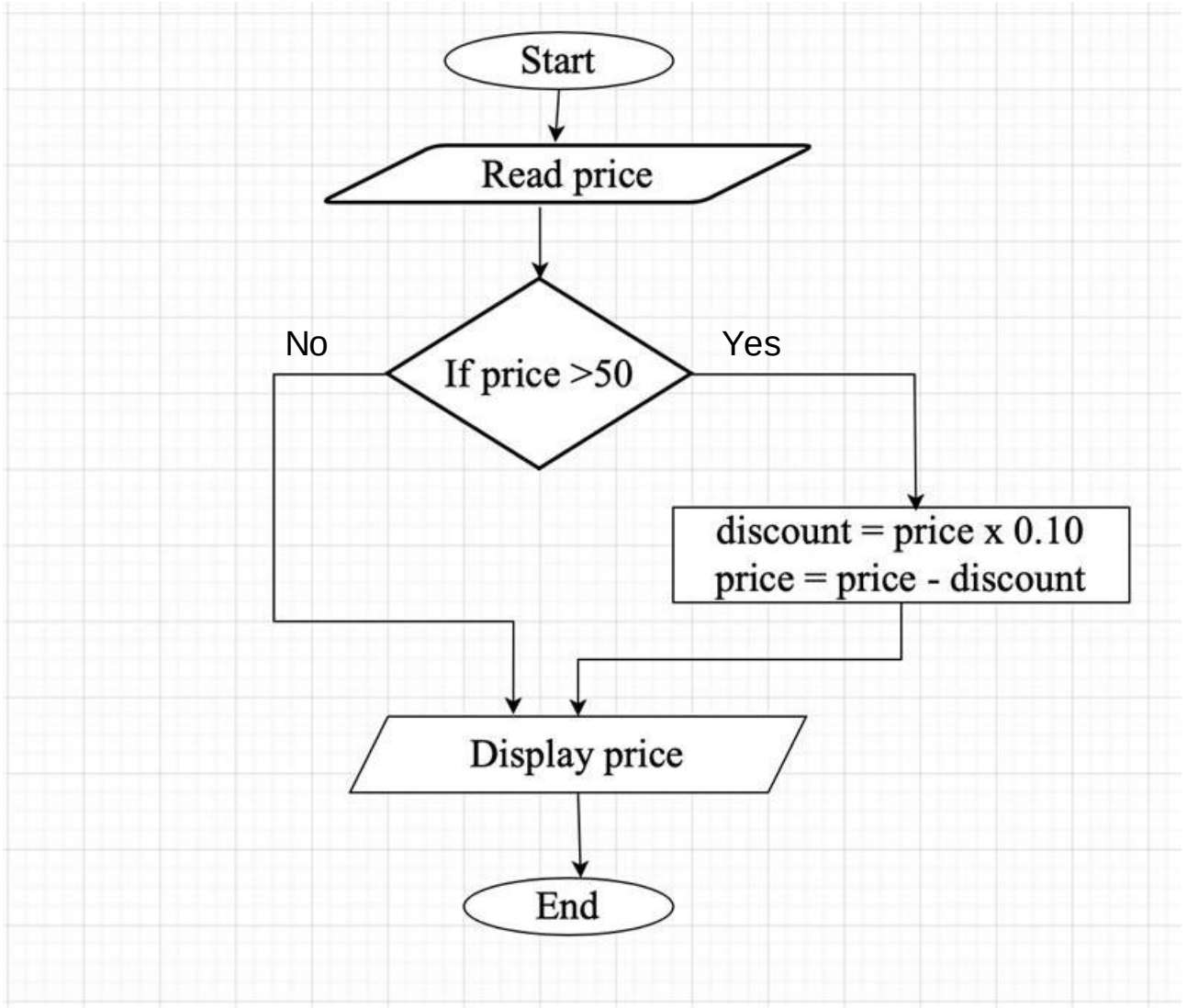
1. Take a frypan and an egg
2. Break and whisk the egg
3. Put oil in the pan
4. Wait until the oil warm up
5. Add the egg
6. Cook it
7. Flip it
8. Plate it up.

You might add spices, maybe a few drops of milk.

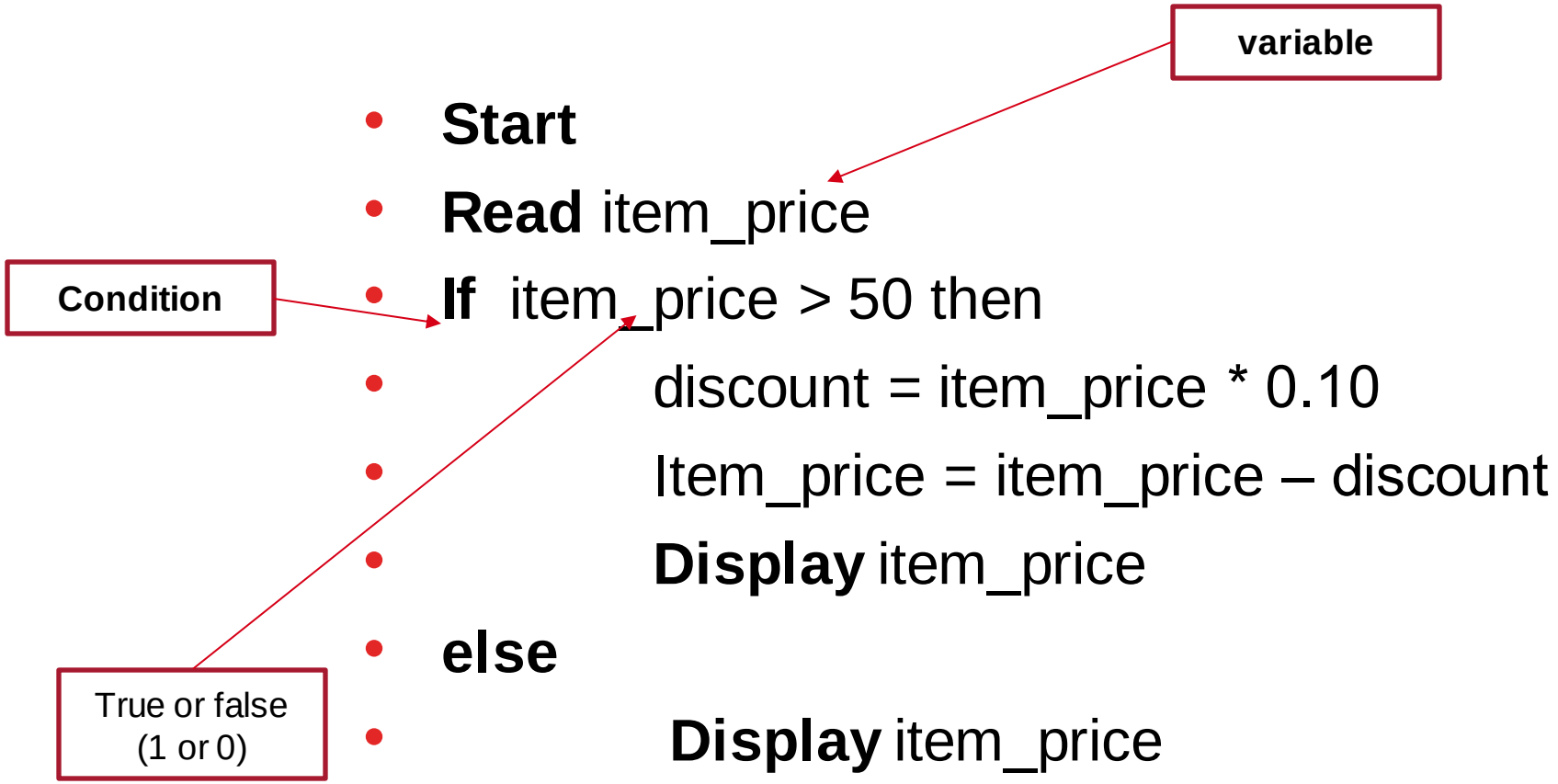
In Class Exercise [Problem Statement]

- Cindy is hired to create program to calculate promotional discount on the items. If the price of the item is greater than $50 then the store will give 10% discount on the item.

In Class Exercise - Flowchart



In Class Exercise - Algorithm



- **Start**
- **Read** item_price
- **If** item_price > 50 then
- discount = item_price * 0.10
- Item_price = item_price – discount
- **Display** item_price
- **else**
- **Display** item_price
- **End**

Example

- Say you have a sock drawer
- write an algorithm to put socks on both feet
  - tip: one foot at a time

Algorithm

```
1   let socks_on_feet = 0
2   while socks_on_feet != 2
3       open sock drawer
4       look for sock
5       if you find a sock then
6           put on sock
7           socks_on_feet++
8           look for matching sock
9           if you find a matching sock then
10              put on matching sock
11              socks_on_feet++
12              close sock drawer
13          else
14              remove first sock from foot
15              socks_on_feet--
16      else
17          do laundry and replenish sock drawer
```

Source: CS50.net

Algorithm

variable

loop

condition

true or false
(1 or 0)

```
1   let socks_on_feet = 0
2   while socks_on_feet != 2
3       open sock drawer
4       look for sock
5       if you find a sock then
6           put on sock
7           socks_on_feet++
8           look for matching sock
9           if you find a matching sock then
10              put on matching sock
11              socks_on_feet++
12              close sock drawer
13          else
14              remove first sock from foot
15              socks_on_feet--
16      else
17          do laundry and replenish sock drawer
```

put 0 into the variable to initialize the value as 0

Source: CS50.net

zyLab – Example 2.1 [Problem Statement]

- Sal's Pizza has hired you to create a calculator program for their upcoming promotional sale.
- They regularly sell a large pizza for $9.99, but are offering a back-to-school discount of 15% for September. Given the number of pizzas to order as input, the program must output the savings amount and subtotal for the pizzas, and then output the total after applying a sales tax of 5%.

zyLab – Exercise 2.1

- Ex. If the input entered is 3
- Then the output would be:

You Saved: $4.50
Subtotal: $25.47
Total due: $26.75

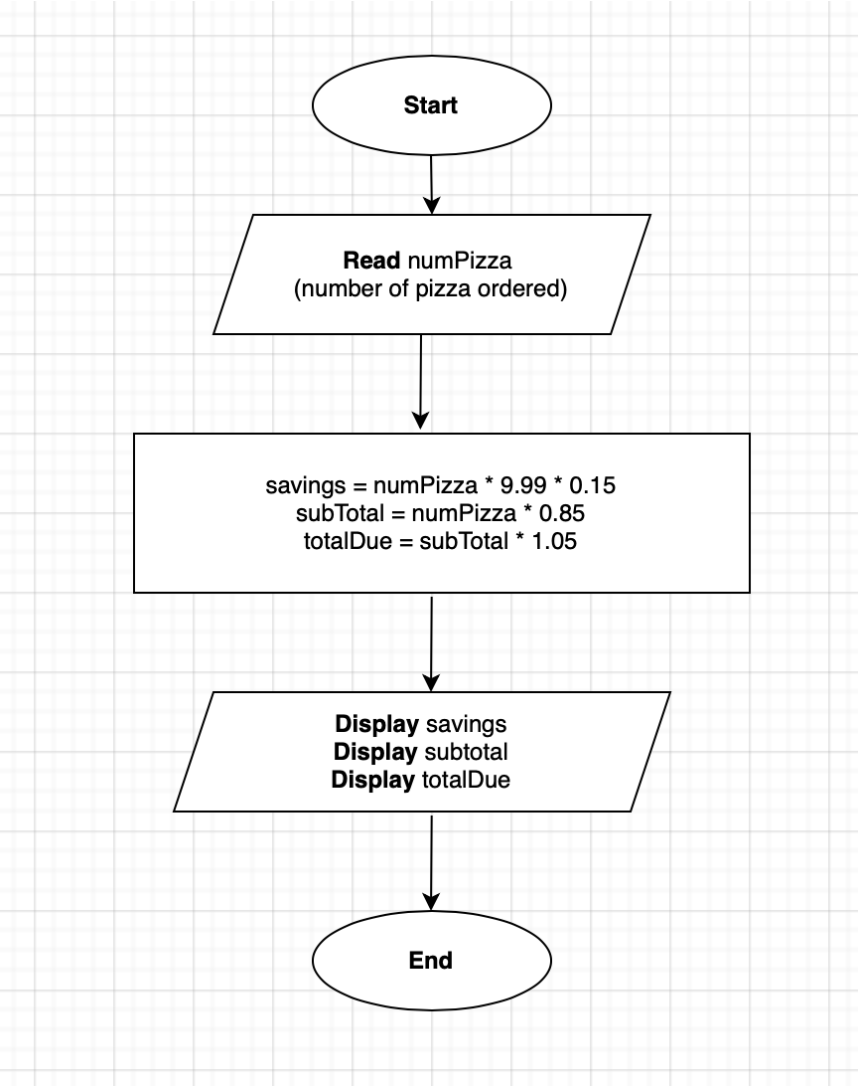
zyLab – Exercise 2.1

- **Task** – given the code below create flowchart for the program specification

```
numPizza = int(input())                 # Read input from user
savings = numPizza * 9.99 * 0.15        # Calculation of savings
subTotal = numPizza * 9.99 * 0.85       # Calculation of subtotal
totalDue = subTotal * 1.05              # Calculation of total with 5% tax

print(f'You Saved: ${savings:.2f}')     # Print savings value formatted to two decimal places
print(f'Subtotal: ${subTotal:.2f}')     # Print subtotal value formatted to two decimal places
print(f'Total due: ${totalDue:.2f}')    # Print total value formatted to two decimal places
```

zyLab – Exercise 2.1 [Flowchart]



zyLab – Exercise 2.2 [Problem Statement]

UNIVERSITY OF CALGARY

- Sal's Pizza would like to sponsor a pizza party for you and your software development team. You decide to design an algorithm that will calculate the amount of pizzas and cost.
- Given the number of people attending a pizza party, output the number of needed pizzas and total cost. For the calculation, assume that people eat 2 slices on average and each pizza has 12 slices and costs $14.95. Round up the number of pizzas so that enough pizzas are ordered.

zyLab – Exercise 2.2 Pizza Party

- Ex. If the input entered is 4
- Then the output would be:
  - Pizzas: 1
  - Cost: $14.95

zyLab – Exercise 2.2 Pizza Party

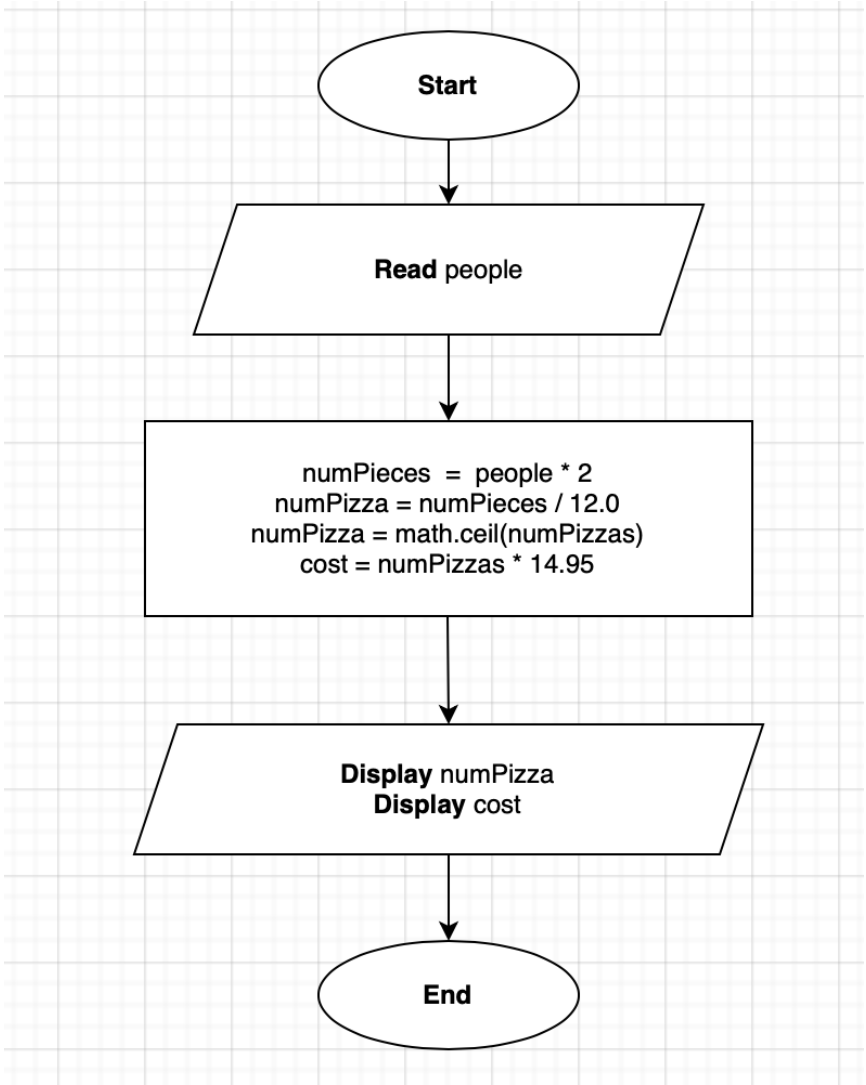
- **TASK**: Write the pseudocode for the algorithm specifications above
- When you are finished, you can compare your answer to the comments in the code on the next slide

zyLab – Exercise 2.2 Pseudocode

UNIVERSITY OF CALGARY

1. Start
2. Prompt the user to enter number of people
3. Read the user input (people)
4. Calculate the number of pieces needed numPieces = people * 2
5. Calculate the number of pizzas needed numPizzas = numPieces / 12.0
6. Round numPizzas upward to its nearest integer
7. Calculate the total cost of the pizzas cost = numPizzas * 14.95
8. Display numPizzas
9. Display cost
10. End

zyLab – Exercise 2.2 [Flowchart]



zyLab – Exercise 2.3 [Problem Statement]

UNIVERSITY OF CALGARY

Note: This exercise is graded.

- An athlete is analyzing their schedule and calculating how much time needs to be spent on nutrition, training, competition, etc. each year. They want to begin with weekly amounts and then calculate how much time is spent per year.
- Below is a program that is a given number of hours, and then prints the equivalent value in seconds and in minutes. It then calculates how many hours would be spent in an entire year (52 weeks).

Exercise 2.3 – Expected output

Input

if input entered is 6

Output

```
Enter the number of hours spent per week.
Number of input hours: 6
Converted to minutes: 360
Converted to seconds: 21600
Expected hours in a year: 312
```

Check your python installation

- Download the hello_world.py from active learning section on D2L
- Copy the hello_world.py in the file that you have before
- Your code files will NOT be saved inside the venv folder, they will be inside your project folder, the venv folder should not be tampered with directly by user
- Active your Virtual Environment
- Run the file from the terminal
- Py hello_world.py
- Python hello_world.py
- Python3 hello_world.py