ENDG 233 – Programming with Data

Branching Logic

Week4: Sept. 27 - Oct 3

Check-in

- Remember to follow the COVID-19 protocols
  - No eating in class, drinking allowed
  - Wear your mask at all times
- Any questions from last week?
- Office hours
- During activities, try to sit and engage with your peers

Portfolio Project FAQs

- Check D2L discussion board for any updates
- For this project, we will not be grading for code optimization
- Remember to use floor division
- Input types
- Deadline: Oct. 1 at 11:59pm
- Any other questions?

Schedule for Week 4

- Examples on branching logic (if/else statements)
- zyLabs exercise based on last week’s material
- Working on portfolio project #1

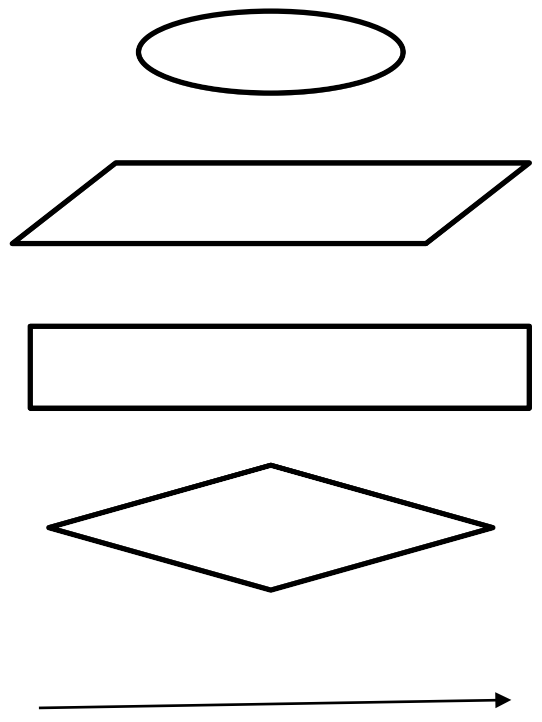
- Coming up on Wednesday:
  - No class!

Portfolio Project FAQs

- Check D2L discussion board for any updates
- For this project, we will not be grading for code optimization
- Remember to use floor division
- Any other questions?

Recap on Week 3

Flowchart symbols



Start / End

Input / Output

Processing / Calculations

Conditional Statements

Direction / connectors

Recap on Week 3

- If-else branches

```
Ex 1: if expression :       Boolean: True or False
            statement

Ex2: if expression:         Boolean: True or False
            statement
        else:
            statement
```

Recap on Week 3

- Multi-branch if-else statement, only one branch will execute

```
Ex 1: if expression:          Boolean: True or False
          statement
      elif expression:        Boolean: True or False
          statement
      elif expression:        Boolean: True or False
          statement
      else:
          statement
```

Recap on Week 3

Operators:

- Equality : ==
- Inequality : !=
- Less than: <
- Greater than: >
- Less than or equal: <=
- Greater than or equal: >=

Recap on Week 3

Logical Operators:

- **AND**: True when both operands are True.
- **OR**: True when at least one operand is True.
- **NOT** (opposite): True when the single operand is False (and False when operand is True).

Recap on Week 3

Order of evaluation: Precedence rules:

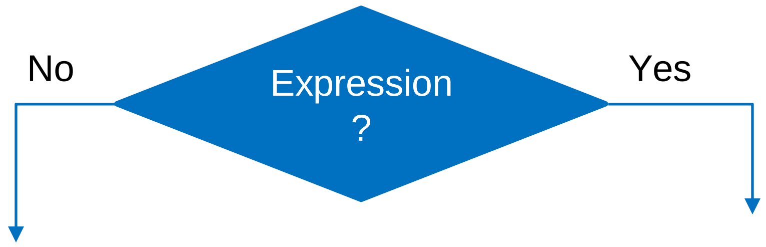
1. **( )**:
2. *** / % + -**: Arithmetic operators (using their precedence rules)
3. <   <=   >   >=   ==   !=
4. Not
5. And
6. Or

Review

- Conditions: decision making based on a expression evaluation
- Expressions evaluate to a value that is:
  - Nonzero (True)
  - Zero (False)

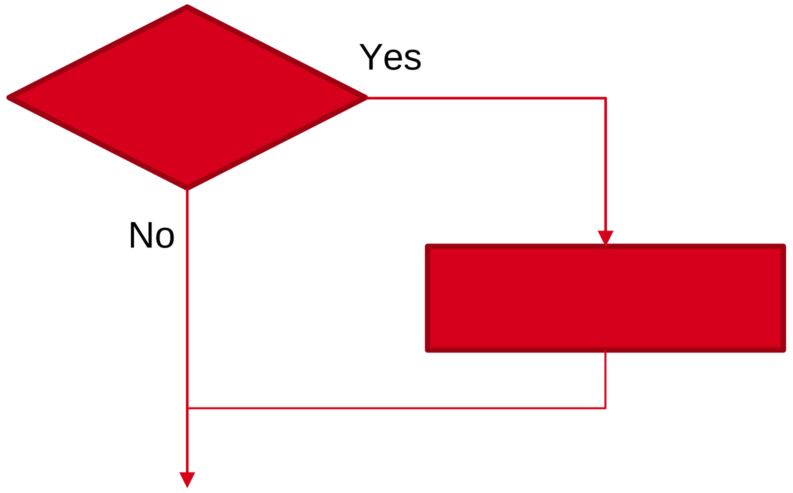
Usually used to combine statements

Relational/comparative: ==, !=, <, ….	Membership operators: in	Identity operator: is	Logical operators: and, or, not
Ex: if (x == y)	Ex: if (x in y)	Ex: if (x is y)	Ex: if (x > y) and (x > z)



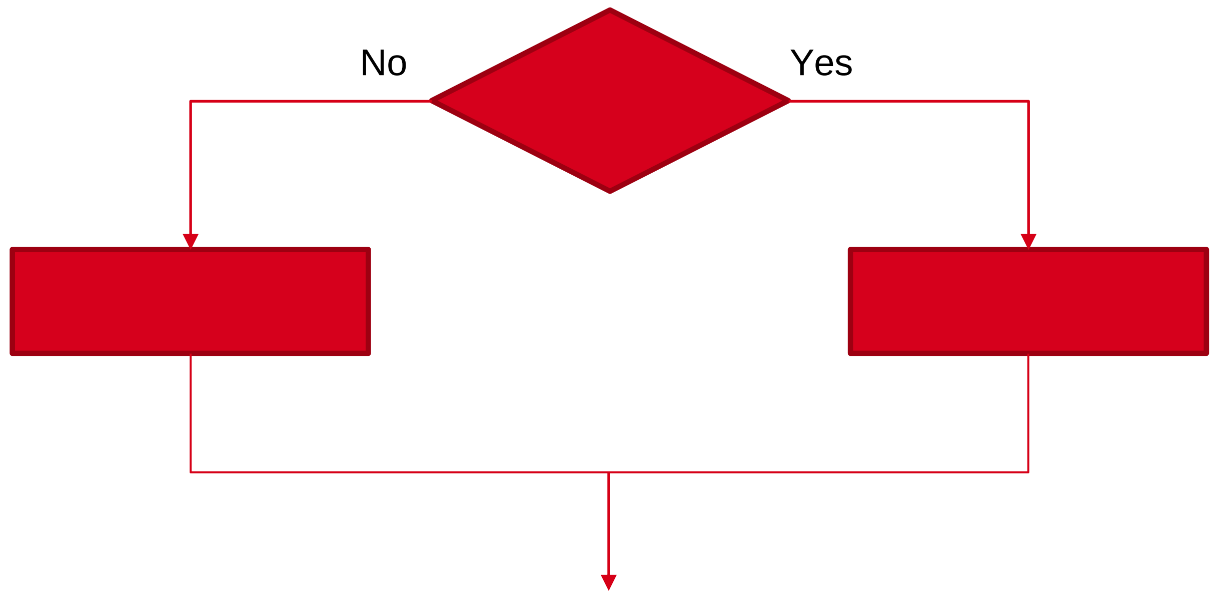
Review

- Use cases:
  - Executing/Skipping code segment: if statement no else



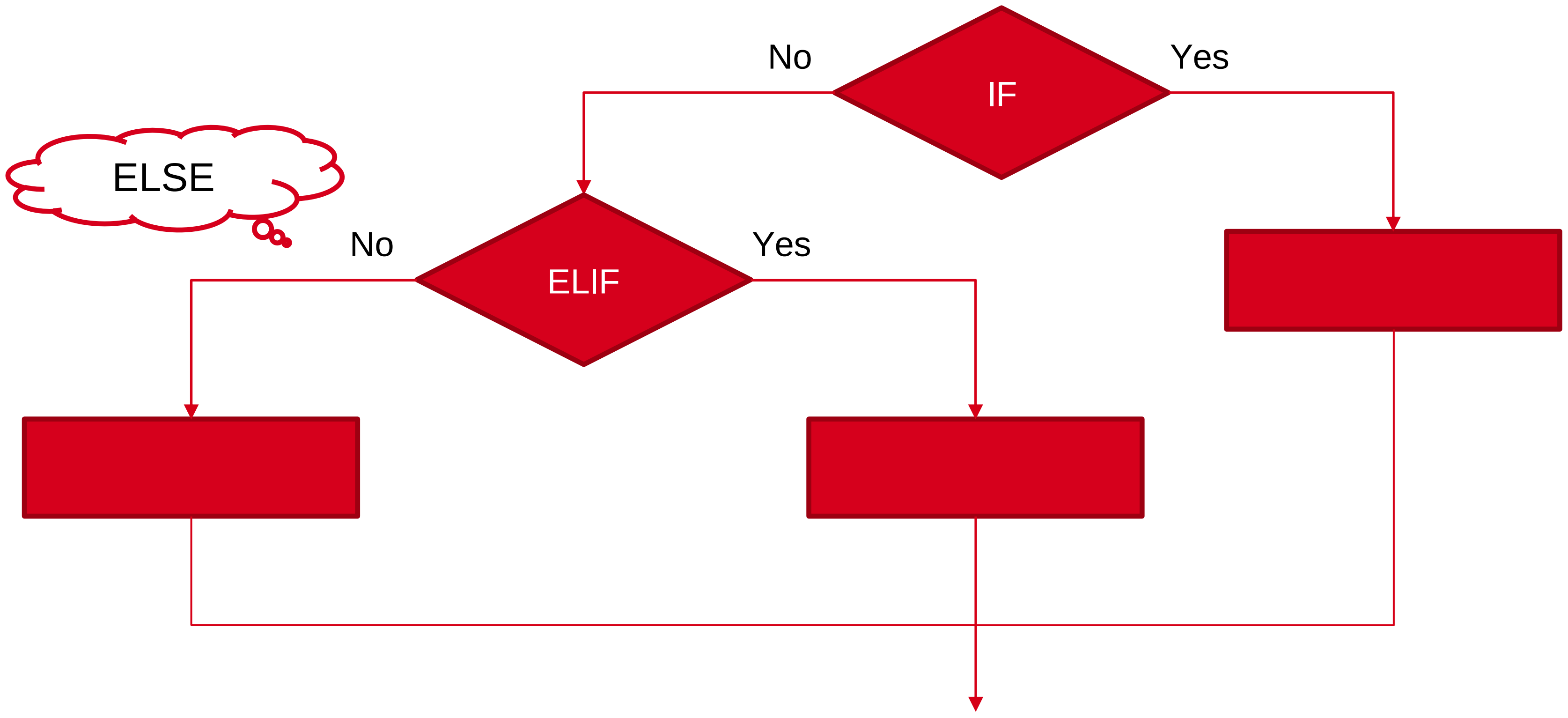
Review

- Use cases:
  - Selecting between two code segments: if-else pair



Review

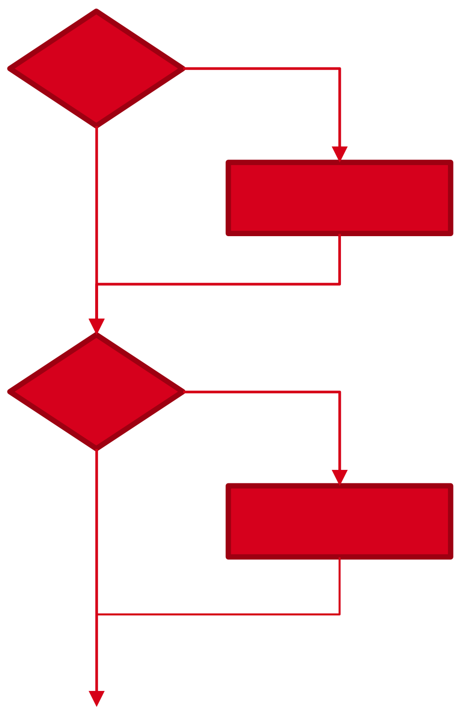
- Use cases:
  - Selecting between multiple branches: if-elif-else



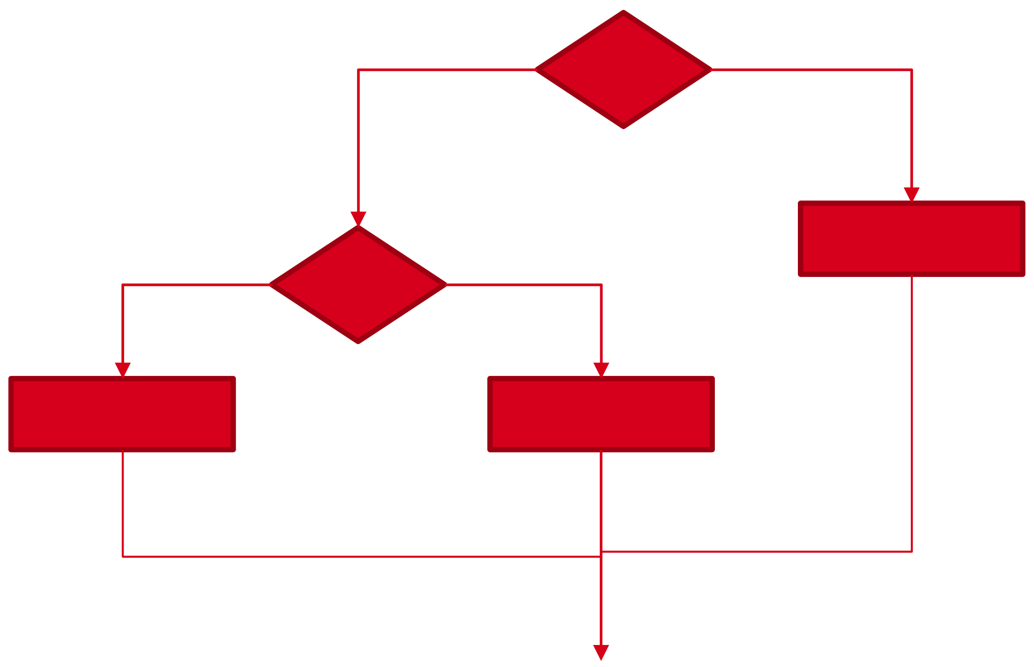
Review

- Multiple if statements vs if-else flow difference

Sequential ifs



If-elif



Review

- Does order of conditions matter?
  - Overlapping statements
    - Given x=60 …
      - Condition 1: X>20
      - Condition 2: X>40
    - What happens if we swap order of conditions ?

Tutorial 4.1 – Max of Three

- Write a program that takes in three integers and outputs the largest value
- Prompt the user with the following message:

```
Please enter the first integer value to compare:
Please enter the second integer value to compare:
Please enter the third integer value to compare:
```

Tutorial 4.1 – Max of Three

- Ex. If the input is:

```
1
2
3
```

- The output is:

```
The maximum value is: 3
```

Tutorial 4.1 – Max of Three

- A sample run of the program is:

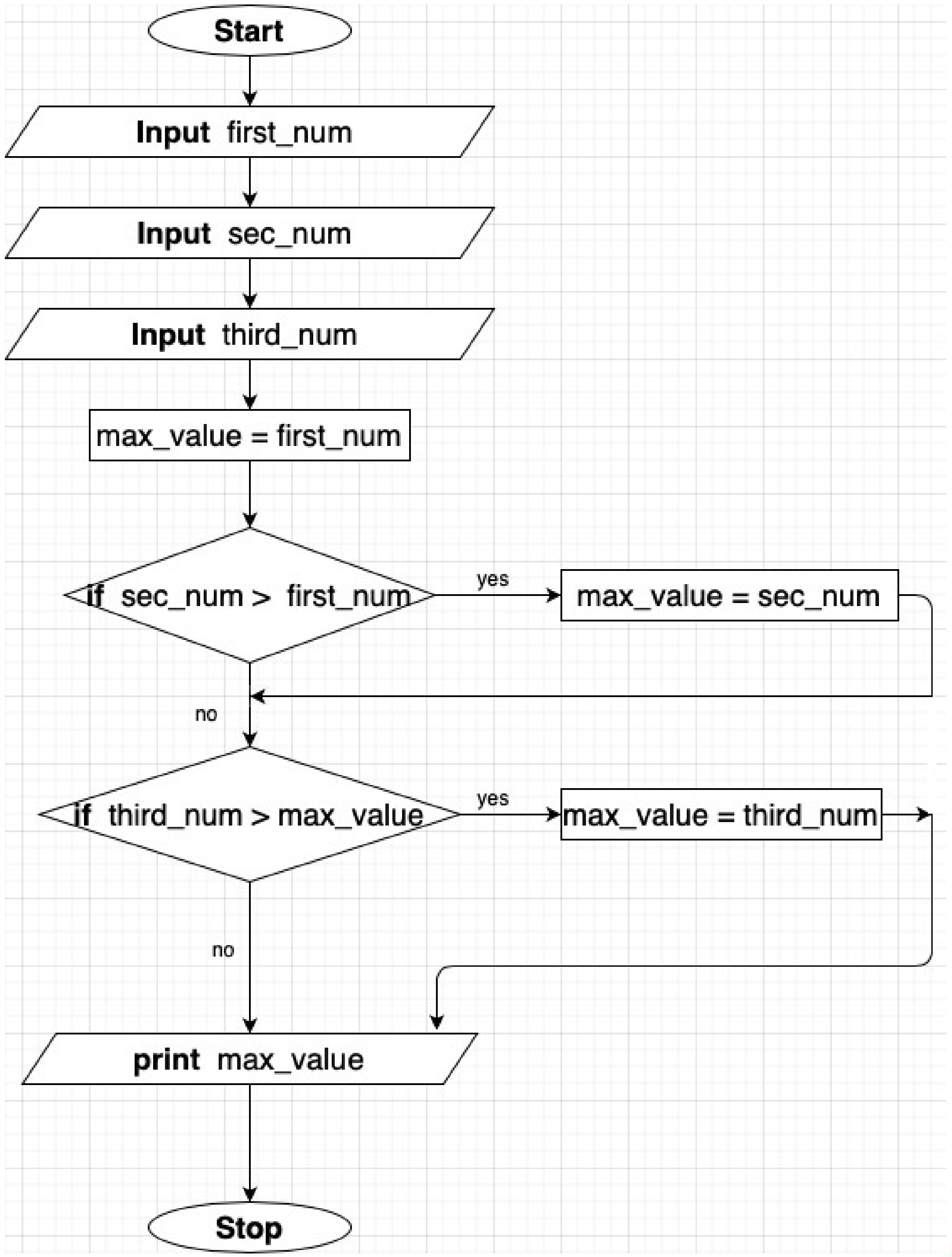
```
Please enter the first integer value to compare:
Please enter the second integer value to compare:
Please enter the third integer value to compare:

The maximum value is: 3
```

Tutorial 4.1 – Max of Three

- **TASK:** Draw the flowchart and/or write a program that finds the maximum of three entered integers

- **HINT:** Remember to start with this message:

```
Please enter the first integer value to compare:
Please enter the second integer value to compare:
Please enter the third integer value to compare:
```


Start
Input first_num
Input sec_num
Input third_num
max_value = first_num
if sec_num > first_num
yes
max_value = sec_num
no
if third_num > max_value
yes
max_value = third_num
no
print max_value
Stop

Tutorial 4.1 – Max of Three

Solution:

```
val1 = int(input('Please enter the first integer value to compare:\n’))
val2 = int(input('Please enter the second integer value to compare:\n’))
val3 = int(input('Please enter the third integer value to compare:\n’))
maxval = val1
if val2 > val1:
        maxval = val2
if val3 > maxval:
        maxval = val3
print('\nThe maximum value is:',maxval)
```

Tutorial 4.2 – Leap Year

- Write a program to determine if the input year is a leap year or not
- The requirements for a given year to be a leap year are:
    - The year must be divisible by 4
    - If the year is a century year (1700, 1800, etc.), the year must be evenly divisible by 400

Tutorial 4.2 – Leap Year

- Write a program that:
  - Prompts the user to input a year using this message:
    - `Please input a year to check if it is a leap year:`
  - Applies the conditional checks and shows the appropriate output
- Ex. 1: If the input is 1712, the output is:

```
1712 is a leap year
```

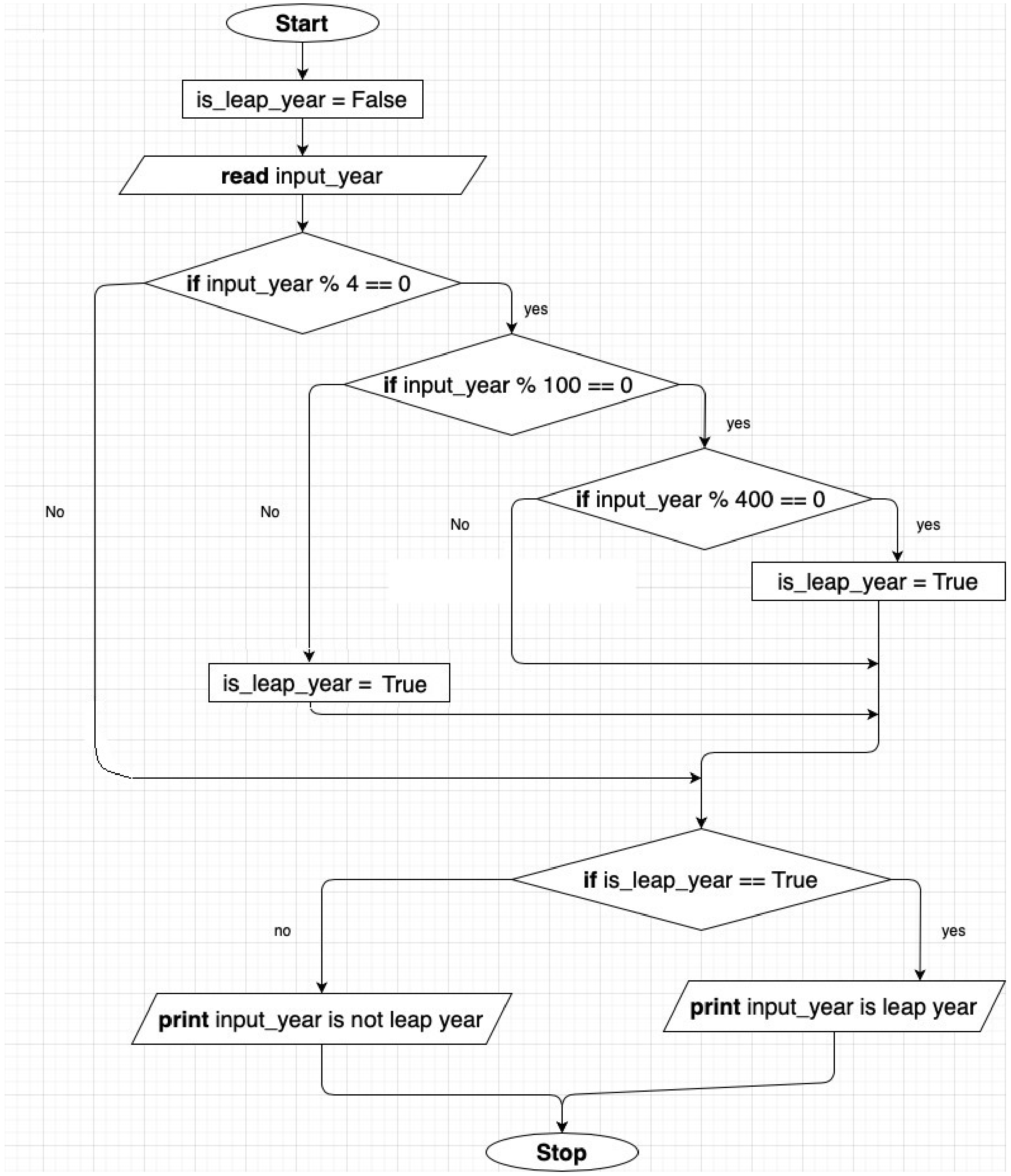
- Ex. 2: If the input is 1913, the output is:

```
1913 is not a leap year
```

Tutorial 4.2 – Leap Year

- **TASK**: Draw the flowchart and/or write a program that determines if the entered year is a leap year or not
- **HINT**: use modulus operator (%)
- Sample output:

```
Please input a year to check if it is a leap year:
1712 is a leap year

Please input a year to check if it is a leap year:
1913 is not a leap year
```


Start
is_leap_year = False
read input_year
if input_year % 4 == 0
yes
if input_year % 100 == 0
yes
if input_year % 400 == 0
yes
is_leap_year = True
No
No
No
is_leap_year = True
if is_leap_year == True
no
yes
print input_year is not leap year
print input_year is leap year
Stop

Tutorial 4.2 – Leap Year

Solution:

```
is_leap_year = False
input_year = int(input('Please input a year to check if it is a leap year:\n'))
if (input_year % 4) == 0:                          # inputYear is divisible by 4
        if (input_year % 100) == 0:                # inputYear is divisible by 100 (century year)
                if (input_year % 400) == 0:        # inputYear is divisible by 400
                        is_leap_year = True
        else:                                      # inputYear is not divisible by 100
                is_leap_year = True
if is_leap_year:
        print(input_year, 'is a leap year’)
else:
        print(input_year, 'is not a leap year')
```

Tutorial 4.3 – Automobile Service Cost

- Write a program to determine the cost of a service at an automobile shop
- The input message for the services menu:

```
Welcome to ENDG_AutoServices. We are happy to provide the
following services:
1. Oil Change        $35
2. Tire Rotation     $19
3. Car Wash          $7

Please enter requested service number:
```

Tutorial 4.3 – Automobile Service Cost

- Upon selecting the service request, an invoice will be displayed to the customer showing the service requested, service cost, and total cost including a 5% GST (2 decimal places)
- For example, if the customer enters 1 for selection, the output will look like this:

```
Service requested:          Oil Change

Service cost:               $35

Total cost:                 $36.75
```

- If the Customer enters a value that is not between 1 and 3 (not a selection option), an error message will be displayed to the costumer: `Service not available`

Tutorial 4.3 – Automobile Service Cost

```
Welcome to ENDG_AutoServices. We are happy to
provide the following services:
1. Oil Change               $35
2. Tire Rotation            $19
3. Car Wash                 $7

Please enter requested service number: 1
Service requested:          Oil Change
Service cost:               $35
Total cost:                 $36.75
```

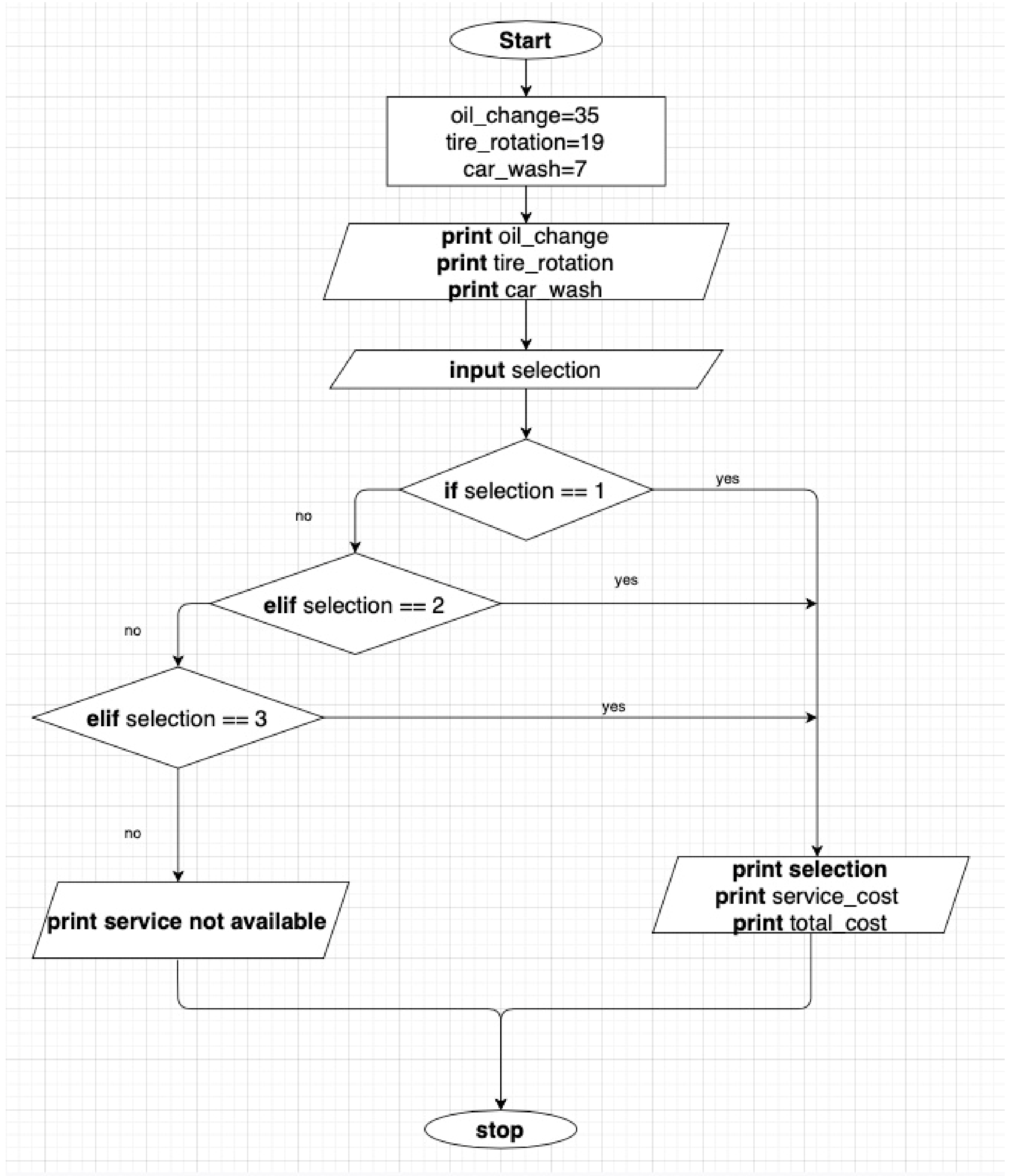
Tutorial 4.3 – Automobile Service Cost

- **Note**: When formatting the menu and invoice strings, to align text we use tabs. In python, you can move the cursor to the beginning of the next tab using the sequence \t (similar to how \n moves the cursor to next line)
- To elaborate, consider this print statement: print('Hi!\t')
- This print statement prints Hi! which takes 3 places in the line, then \t advances the cursor 5 spaces to align with the beginning of the next tab in line
- If we modify the print statement to: print('Hi!\tHi!') The output will be: Hi!   Hi!
- **Make sure to match the tabs provided in the output sample!**

Tutorial 4.3 – Automobile Service Cost

- **TASK**: Draw a flowchart and/or write code to print the menu, requesting the service selection, then print the invoice message with the selection, cost and total cost (2 decimal places) for the customer (or print the error message if the selection is invalid)
- **HINT**: Start with this code:

```
oil_change = 35

tire_rotation = 19

car_wash = 7
```


Start
oil_change=35
tire_rotation=19
car_wash=7
print oil_change
print tire_rotation
print car_wash
input selection
if selection == 1
yes
no
elif selection == 2
yes
no
elif selection == 3
yes
no
print service not available
print selection
print service_cost
print total_cost
stop

Tutorial 4.3 – Automobile Service Cost

```
oil_change = 35
tire_rotation = 19
car_wash = 7
print('Welcome to ENDG_AutoServices. We are happy to provide the
following services:\n’)
print(f'1. Oil Change\t\t${oil_change}’)
print(f'2. Tire Rotation\t${tire_rotation}’)
print(f'3. Car Wash\t\t${car_wash}’)
print()
selection = int(input('Please enter requested service number:\n'))
```

Tutorial 4.3 – Automobile Service Cost

```
if selection == 1:
        print('Service requested:\tOil Change')
        print(f'Service cost:\t\t${oil_change}')
        print(f'Total cost:\t\t${oil_change*1.05:.2f}’)
elif selection == 2:
        print('Service requested:\tTire Rotation')
        print(f'Service cost:\t\t${tire_rotation}')
        print(f'Total cost:\t\t${tire_rotation*1.05:.2f}’)
elif selection == 3:
        print('Service requested:\tCar Wash')
        print(f'Service cost:\t\t${car_wash}')
        print(f'Total cost:\t\t${car_wash*1.05:.2f}’)
else:
        print('Service not available')
```

Notes on formatting output

- The single quotation symbol marks the beginning and ending of a string. To indicate in a string that the quotation mark is to be printed and not denote that start or end of a string, we use the escape sequence

```
\'
```

- For example, to print the word `student's`, the print statement will be:

```
print('student\'s')
```

Notes on formatting output

- When formatting a string to show in somewhat a tabular view, to align text we use `tabs`.
- In python, a tab starts every 8 places in a line, you can move the cursor to the beginning of next tab using the escape sequence

  `\t`

- This is similar to how `\n` moves the cursor to next line.

Notes on formatting output

- To elaborate, consider this print statement:

```
print('Hi!\t')
```

- This print statement prints

```
Hi!
```

- Which takes 3 places in the line, then `\t` advances the cursor 5 spaces to align with the beginning of the next tab in line. If we modify the print statement to:

```
print('Hi!\tHi!')
```

- The output will be

```
Hi!     Hi!
```

Notes on formatting output

- Note that a tab is not equivalent to spaces when compared directly.
- For example.

  `Hi!`*(space)(space)(space)(space)(space)*`Hi!`

- `Is NOT equivalent to`

  `Hi!`*(tab)*`Hi!`