CSC101 – Introduction to ICT
Lecture 06
Selection

Course Instructors

Dr. Majid Iqbal **Dr. Ashfaq Hussain Farooqi** **Ms. Sajida Kalsoom** **Mr. Rizwan Rashid**

Lecture Outline

- Introduction
- Boolean Types, Values, and Expressions
- One-way **if** statement
- Two-way **ifelse** statement
- Nested **if** statement
- Multi-way **if-elif-else** statement
- Logical Operators
- Conditional expressions

Introduction

- Control structure: logical design that controls order in which set of statements execute
- Sequence structure: set of statements that execute in the order they appear
- Decision structure: specific action(s) performed only if a condition exists
  - Also known as selection structure

Introduction

```
1. name = input('What is your name? ')
2. age = int(input('What is your age? '))
3. print('Here is the data you entered:')
4. print('Name:', name)
5. print('Age:', age)
```

Introduction

- Control structure: logical design that controls order in which set of statements execute
- Sequence structure: set of statements that execute in the order they appear
- Decision structure: specific action(s) performed only if a condition exists
  - Also known as selection structure

Boolean Types, Values, and Expressions

- Selection statements use conditions, which are *Boolean expressions.*
- *A Boolean expression is an expression that evaluates to a Boolean value* **True** *or* **False**.
- *Python provides six comparison operators (also known as relational operators)*

TABLE 4.1 Comparison Operators

Python Operator	*Mathematics Symbol*	*Name*	*Example* (radius is 5)	*Result*
`<`	$<$	less than	`radius < 0`	`False`
`<=`	$\leq$	less than or equal to	`radius <= 0`	`False`
`>`	$>$	greater than	`radius > 0`	`True`
`>=`	$\geq$	greater than or equal to	`radius >= 0`	`True`
`==`	$=$	equal to	`radius == 0`	`False`
`!=`	$\neq$	not equal to	`radius != 0`	`True`

Boolean Types, Values, and Expressions

- A variable that holds a Boolean value (True/False) is known as a **Boolean variable**.
- The **Boolean data type** is used to represent Boolean values. E.g. **lightsOn = True**
  - Can be verified using **Type(lightsOn)** function → <class 'bool'>
- **True** and **False** are literals, like 10, and are reserve words
- Internally, Python uses **1** to represent **True** and **0** for **False**.
  - You can use the **int** function to convert a Boolean value to an integer.
  - For example, print(int(**True**)) displays **1** and print(int(**False**)) displays **0**

Conditional processing

- Python has several types of selection statements:
  - One-way **if** statements
  - Two-way **if-else** statements
  - Nested **if** statements
  - Multi-way **if-elif-else** statements
  - Conditional expressions

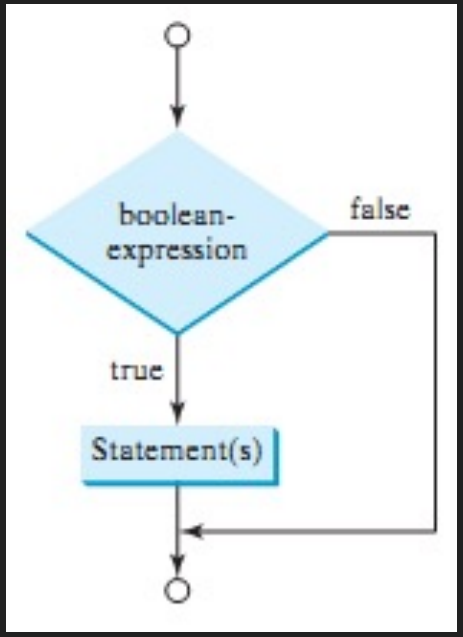
if Statement

- A one-way **if** statement executes an action if and only if the condition is true.
- Syntax

```
if boolean-expression:
    statement(s)
```

- `Statement(s)` within if must be indented

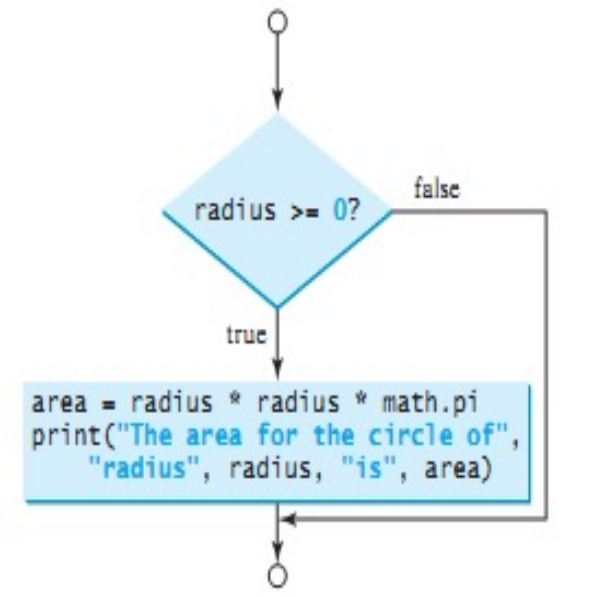
The symbol diamond shows Boolean-expression.



if Statements - Example

```
if radius >= 0:
    area = radius * radius * math.pi
    print("The area for the circle of radius", radius, "is", area)
```

Indented Block



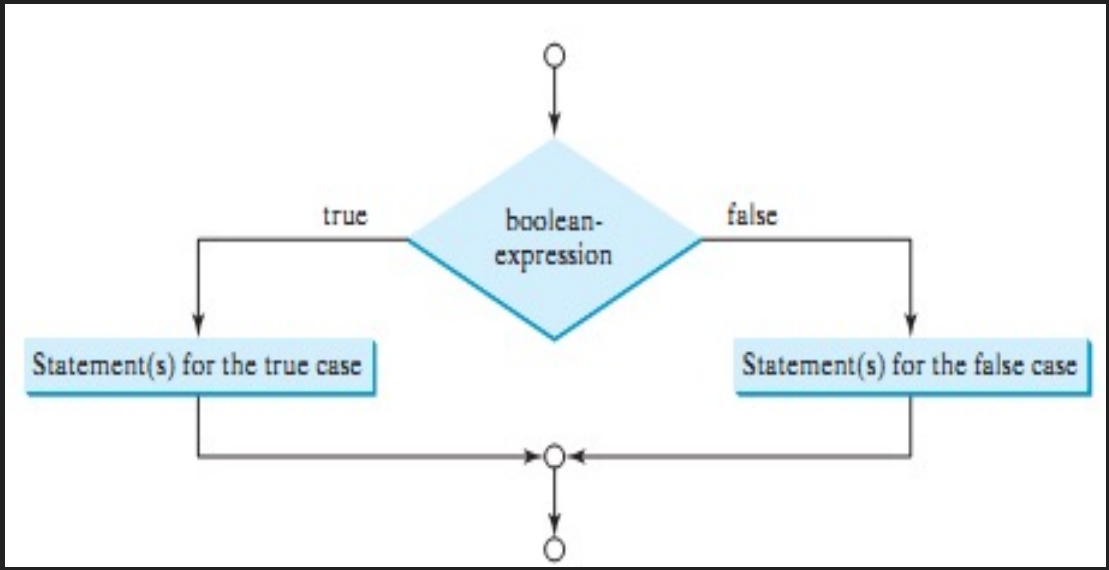
- **Increase indent** after an **if** statement
- **Always maintain** the indent to make statement(s) part of **if block**
- **Each statement** must be indented using the **same number of spaces**

```
1  if radius >= 0:
2      area = radius * radius * math.pi  # Compute area
3    print("The area for the circle of radius", radius, "is", area)
```

Two-Way **if-else** Statements

- A two-way **if-else statement** decides which statements to execute based on whether the condition is **true** or **false**
- Two possible paths: One in case condition is **true** and other when condition is **false**
- **Syntax**

```
if boolean-expression:
    statement(s)-for-the-true-case
else:
    statement(s)-for-the-false-case
```



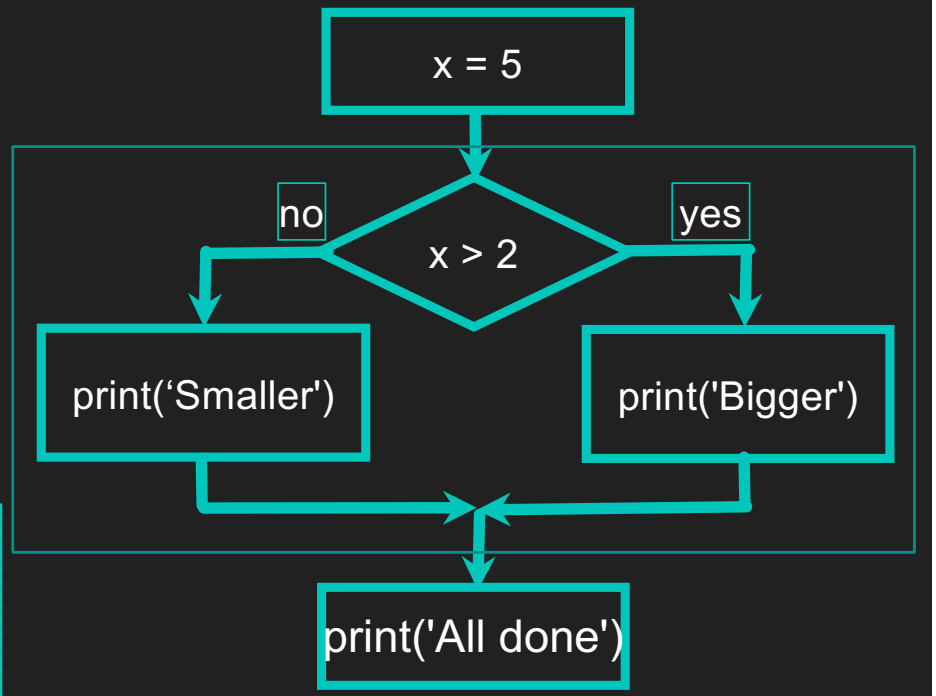
Two-Way **if-else** Statement – Example

Example-1

```
x = 5

if x > 2 :
    print('Bigger')
else :
    print('Smaller')

print('All done')
```



Example-2

```
if radius >= 0:
    area = radius * radius * math.pi
    print("The area for the circle of radius", radius, "is", area)
else:
    print("Negative input")
```

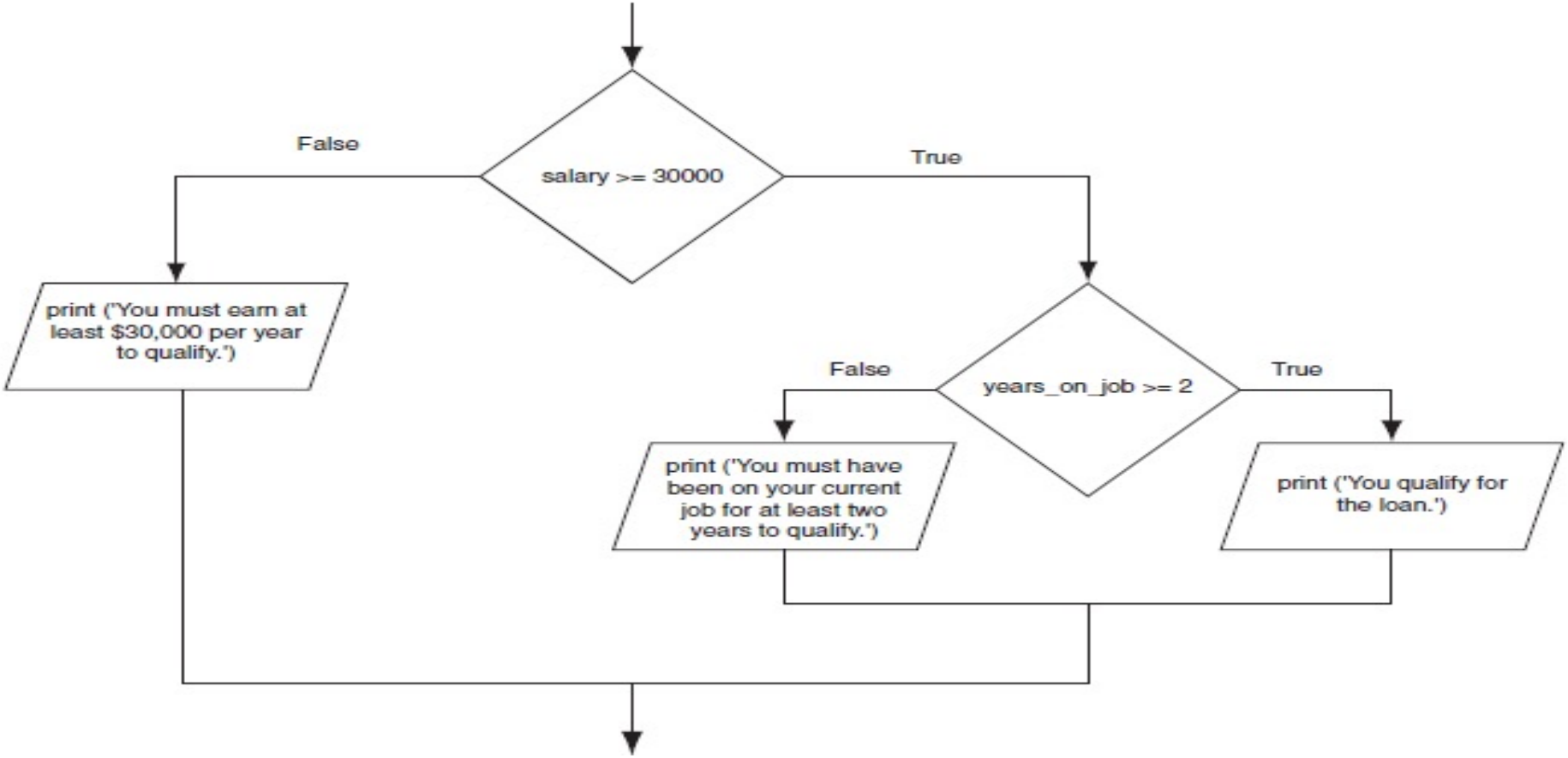
Two-Way **if-else** Statement – Example

- Chris owns an auto repair business and has several employees. If any employee works over 40 hours in a week, he pays them 1.5 times their regular hourly pay rate for all hours over 40. He has asked you to design a simple payroll program that calculates an employee's gross pay, including any overtime wages
- Algorithm
  - Get the number of hours worked.
  - Get the hourly pay rate.
  - If the employee worked more than 40 hours:
    - Calculate and display the gross pay with overtime.
  - Else:
    - Calculate and display the gross pay as usual

Nested Decision Structures

- A decision structure can be nested inside another decision structure
  - Commonly needed in programs
- **Example:**
  - Determine if someone qualifies for a loan, they must meet two conditions:
    - Must earn at least $30,000/year
    - Must have been employed for at least two years
  - Check first condition, and if it is true, check second condition

Nested Decision Structures – Example



Nested Decision Structures – Example

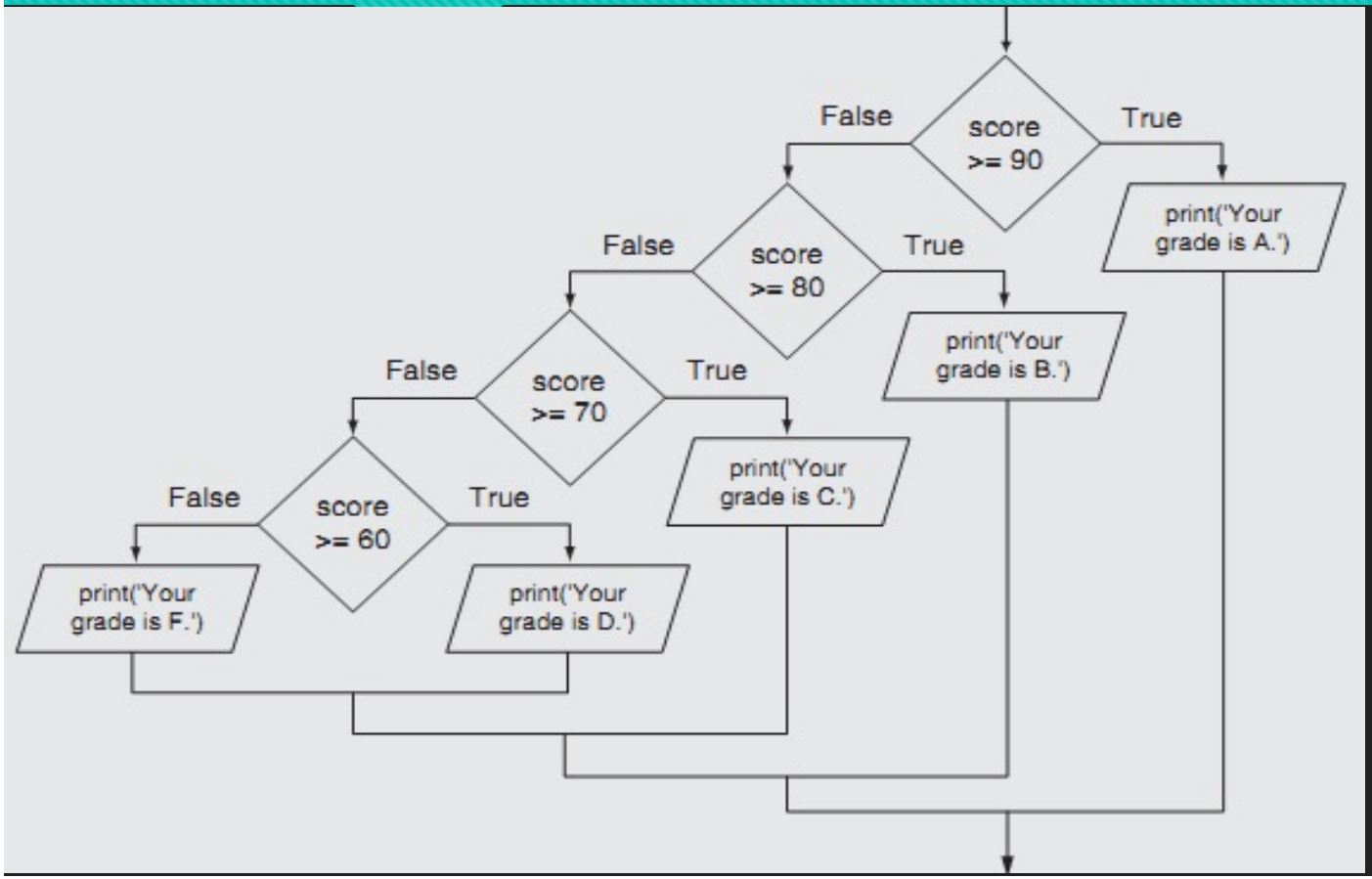
Dr. Suarez teaches a literature class and uses the following 10 point grading scale for all of his exams:

Test Score	Grade
90 and above	A
80–89	B
70–79	C
60–69	D
Below 60	F

1. Ask the user to enter a test score.
2. Determine the grade in the following manner:

If the score is greater than or equal to 90, then the grade is A.
Else, if the score is greater than or equal to 80, then the grade is B.
Else, if the score is greater than or equal to 70, then the grade is C.
Else, if the score is greater than or equal to 60, then the grade is D.
Else, the grade is F.

Nested Decision Structures – Example



Python-Code Snippet

```
# Determine the grade.
if score >= A_SCORE:
    print('Your grade is A.')
else:
    if score >= B_SCORE:
        print('Your grade is B.')
    else:
        if score >= C_SCORE:
            print('Your grade is C.')
        else:
            if score >= D_SCORE:
                print('Your grade is D.')
            else:
                print('Your grade is F.')
```

The if-elif-else Statement

- **Syntax**

```
if boolean-expression-1 :
    statement(s)
elif boolean-expression-2 :
    statement (s)
#Insert as many elif clauses as necessary. . .
else:
    statement (s)
```

The if-elif-else Statement – Example

Nested if-Structure

```
# Determine the grade.
if score >= A_SCORE:
    print('Your grade is A.')
else:
    if score >= B_SCORE:
        print('Your grade is B.')
    else:
        if score >= C_SCORE:
            print('Your grade is C.')
        else:
            if score >= D_SCORE:
                print('Your grade is D.')
            else:
                print('Your grade is F.')
```

If-elif-else

```
if score >= A_SCORE:
    print('Your grade is A.')
elif score >= B_SCORE:
    print('Your grade is B.')
elif score >= C_SCORE:
    print('Your grade is C.')
elif score >= D_SCORE:
    print('Your grade is D.')
else:
    print('Your grade is F.')
```

Logical Operators

- *Logical operators*, also known as *Boolean operators*, operate on Boolean values to create a new Boolean value

Operator	Meaning
and	The and operator connects two Boolean expressions into one compound expression. Both subexpressions must be true for the compound expression to be true.
or	The or operator connects two Boolean expressions into one compound expression. One or both subexpressions must be true for the compound expression to be true. It is only necessary for one of the subexpressions to be true, and it does not matter which.
not	The not operator is a unary operator, meaning it works with only one operand. The operand must be a Boolean expression. The not operator reverses the truth of its operand. If it is applied to an expression that is true, the operator returns false. If it is applied to an expression that is false, the operator returns true.

Logical Operators

TABLE 4.4 Truth Table for Operator `not`

p	not p	*Example (assume* `age = 24, gender = 'F'`*)*
`True`	`False`	`not (age > 18)` is `False`, because `(age > 18)` is `True`.
`False`	`True`	`not (gender == 'M')` is `True`, because `(gender == 'M')` is `False`.

TABLE 4.5 Truth Table for Operator `and`

p_1	p_2	p_1 and p_2	*Example (assume* `age = 24, gender = 'F'`*)*
`False`	`False`	`False`	`(age > 18) and (gender == 'F')` is `True`, because `(age > 18)` and `(gender == 'F')` are both `True`.
`False`	`True`	`False`	
`True`	`False`	`False`	`(age > 18) and (gender != 'F')` is `False`, because `(gender != 'F')` is `False`.
`True`	`True`	`True`	

TABLE 4.6 Truth Table for Operator `or`

p_1	p_2	p_1 or p_2	*Example (assume* `age = 24, gender = 'F'`*)*
`False`	`False`	`False`	`(age > 34) or (gender == 'F')` is `True`, because `(gender == 'F')` is `True`.
`False`	`True`	`True`	
`True`	`False`	`True`	`(age > 34) or (gender == 'M')` is `False`, because `(age > 34)` and `(gender == 'M')` are both `False`.
`True`	`True`	`True`	

Logical Operators – Example

Program checks whether a number is divisible by 2 and 3, by 2 or 3, and by 2 or 3 but not both

```
# Receive an input
number = eval(input("Enter an integer: "))

if number % 2 == 0 and number % 3 == 0:
    print(number, "is divisible by 2 and 3")

if number % 2 == 0 or number % 3 == 0:
    print(number, "is divisible by 2 or 3")

if (number % 2 == 0 or number % 3 == 0) and  \
      not (number % 2 == 0 and number % 3 == 0):
    print(number, "is divisible by 2 or 3, but not both")
```

Checking Numeric Ranges with Logical Operators

- To determine whether a numeric value is within a specific range of values, use **`and`**
  - Example: `x >= 10 and x <= 20`
- To determine whether a numeric value is outside of a specific range of values, use **`or`**
  - Example: `x < 10 or x > 20`

Conditional Expressions

A conditional expression evaluates an expression based on a condition.

- Conditional expressions are in a completely different style.
- **Syntax :**

  Expression-1 **if** boolean-expression **else** expression2

Example-1

```
if x > 0:
    y = 1
else:
    y = -1
```

Can be written as

```
y = 1 if x > 0 else -1
```

MaxNum-Example

```
max = number1 if number1 > number2 else number2
```

Summary

- Boolean Expression
- Conditional Statements
- Logical Operators
- Conditional Expression

Next Lecture

- While Loop
- Working with While Loop
- For Loop
- Range() Function