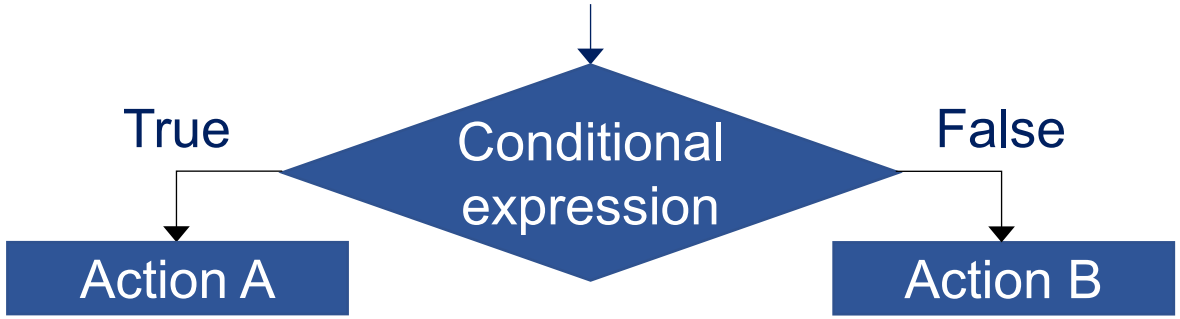
Boolean, branching, Advanced assignment, import

Boolean Data Type



- In most computer programming languages, a **Boolean** data type is a data type with only two possible values, either **True** or **False**.

- Conditional expression, also called **Boolean expression, may be composed of a combination of the Boolean constants True or False, Boolean-typed variables, Boolean-valued operators, and Boolean-valued functions**.

Relational Operators

A **relational operator** compares two numbers (**float** or **int**) and returns a **Boolean** value of either **True** or **False**.

Relational Operator	Meaning	Example
==	equal to	a == 1
!=	not equal to	b != 2
<	less than	c < 3
<=	less than or equal to	d <= 4
>	greater than	f > 5.0
>=	greater than or equal to	f >= 6.0

Logical/ Boolean Operators

Logical operators connect Boolean values and expressions and **return** a Boolean value as a result.

A	B	not A	A and B	A or B
False	False	True	False	False
False	True	True	False	True
True	False	False	False	True
True	True	False	True	True

Operator	Example	Meaning
not	**not** num < 0	**Flip T/ F**
and	(num1 > num2) **and** (num2 > num3)	**Return True only if both are True**
or	(num1 > num2) **or** (num2 > num3)	**Return True if either one is True**

Summary

IF-ELSE

```
if condition:
    indentedStatementBlockForTrueCondition
else:
    indentedStatementBlockForFalseCondition
```

IF-ELIF-ELSE

```
if expression1:
    suite1
elif expression2:
    suite2
else:
    suite3
```

More on Selection (Branching) SYNTAX

(No NEW syntax)

Nested IF

```
if condition1:
    if condition2:
        SUITE A
    else:
        SUITE B
else:
    SUITE C
```

slido

Please download and install the Slido app on all computers you use

What will be the output of the following code?

ⓘ Start presenting to display the poll results on this slide.

```
x = 5
if x < 10:
    print("Less than 10")
else:
    print("10 or more")
```

Less than 10

slido

Please download and install the Slido app on all computers you use

What is the output of the following code?

ⓘ Start presenting to display the poll results on this slide.

```
x = 20
if x < 10:
    print("Less than 10")
elif x == 20:
    print("Equal to 20")
else:
    print("Greater than 10 but not 20")
```

Equal to 20

slido

What is the output of the code ?

ⓘ Start presenting to display the poll results on this slide.

```python
if (7 < 0) and (0 < -9):
    print("Python")
elif False or (100 > 0):
    print("good")
else:
    print("bad")
```

good

slido

What is the output of the code ?

ⓘ Start presenting to display the poll results on this slide.

```
x = 0
a = 3
b = -1
if a > 0:
    if b < 0:
        x = x + 5
    elif a > 5:
        x = x + 4
    else:
        x = x + 3
else:
    x = x + 2
print(x)
```

5

Advanced topics:**SHORT CIRCUIT EVALUATION**

slido

What is the output of the following code?

```
print(3/0 > 3 and 3 < 0)
```

ⓘ Start presenting to display the poll results on this slide.

```
print(3/0 > 3 and 3 < 0)
```

```
print(3/0 > 3 and 3 < 0)
ZeroDivisionError: division by zero
```

slido

What is the output of the following code?

```
print(3 < 3 and 3/0 > 3)
```

ⓘ Start presenting to display the poll results on this slide.

```
print(3 < 3 and 3/0 > 3)
```

False

slido

What is the output of the following code?

```
print(3 > 0 or 3/0 != 1)
```

ⓘ Start presenting to display the poll results on this slide.

```
print(3 > 0 or 3/0 != 1)
```

True

SHORT CIRCUIT EVALUATION

- Both the *and* and *or* operators perform short-circuit evaluation.
- If the expression on the left side of the *and* operator is false, the expression on the right side will not be checked. Because the compound expression will be false if only one of the subexpressions is false, it would waste CPU time to check the remaining expression. So, when the and operator finds that the expression on its left is false, it short-circuits and does not evaluate the expression on its right.
- If the expression on the left side of the *or* operator is true, the expression on the right side will not be checked. Because the compound expression will be true if only one of the subexpressions is true, it would waste CPU time to check the remaining expression. So, when the or operator finds that the expression on its left is true, it short-circuits and does not evaluate the expression on its right.

Avoiding AttributeError (e.g., accessing None)

```python
# Safe access using short-circuit
if user is not None and user.get("is_active"):
    print("User is active")
```

AVOIDING DIVISION BY ZERO

```
# Safe division
if denominator != 0 and (10 / denominator) > 1:
    print("Result is greater than 1")
```

EFFICIENT EVALUATION (SKIP EXPENSIVE FUNCTIONS)

```
if fast_condition or expensive_check():
    print("Condition passed")
```

REAL-LIFE EXAMPLES WHERE SHORT-CIRCUITING IS BENEFICIAL

Summary Table

Scenario	Expression	Benefit
Avoid `NullPointerException`	`obj != null && obj.method()`	Prevents runtime crash
Input validation	`x > 0 && x < 100 && check(x)`	Skip costly check early
File existence before read	`exists(file) && read(file)`	Prevent file error
Loop guard	`data != null && !data.empty()`	Safe loop condition
Efficient logging	`debug && compute_log()`	Skip expensive call when unneeded

QUESTION (PSEUDO CODE STYLE)

- **Problem:**
  A company only grants access to its secure system if the user is **logged in** and has **admin privileges**.
  Write a pseudo code that checks if access should be granted using **short-circuit logic**.

```
IF user_is_logged_in AND user_is_admin THEN
    PRINT "Access Granted"
ELSE
    PRINT "Access Denied"
```

Advanced topics

1. CHAINED ASSIGNMENT

- We want more variables to take the same value:
- **a = b = 1**
- **a = b = c = 10**

- Don’t make it too long or clumsy…
- Readability!!!

slido

What is the output of the following python code?

ⓘ Start presenting to display the poll results on this slide.

```
a = b = c = 10
a = a - 5
a = b
b = a
print (a + b + c)
```

30

2. MULTIPLE ASSIGNMENT

- more than assign the result of a single expression to a single/multiple variables. (1:1, 1:n)
  - assigning multiple variables at one time.
- The left and right side must have the same number of elements. (n:n)
- Use comma for multiple assignment

  a , b = 10, 20

- Note: It supports more than two elements

  a , b , c = 10 , 11 , 12

- Make sure *same* number of elements on LHS and RHS

slido

What is the output of the following python code?

ⓘ Start presenting to display the poll results on this slide.

```
a,b = 1,2
a=b
b=a
print("First: ",a,b, end =" vs ")
a,b = 1,2
a,b = b,a
print("Second: ",a,b)
```

First: 2 2 vs Second: 2 1

SWAPPING TWO VALUES

- One standard way in many programming languages is to use a temporary variable as a buffer:

```
tmp = a
a   = b
b   = tmp
```

We can then swap the reference (computationally)

IMPORT

Standard Library Functions and the `import` Statement

```
import math
radiusString = input("Enter the radius of your circle:")
radiusFloat = float (radiusString)
circumference = 2 * math.pi * radiusFloat
area = math.pi * radiusFloat * radiusFloat
```

Standard Library Functions and the `import` Statement (cont'd.)

- Modules: files that stores functions of the standard library
  - Help organize library functions not built into the interpreter
  - Copied to computer when you install Python
- To call a function stored in a module, need to write an `import` statement
  - Written at the top of the program
  - Format: `import` *`module_name`*
- Dot notation: notation for calling a function belonging to a module
  - Format: `module_name.function_name()`

The `math` Module

- `math` module: part of standard library that contains functions that are useful for performing mathematical calculations
  - Typically accept one or more values as arguments, perform mathematical operation, and return the result
  - Use of module requires an `import math` statement

The `math` Module (cont'd.)

Table 5-2 Many of the functions in the `math` module

`math` Module Function	Description
`acos(x)`	Returns the arc cosine of `x`, in radians.
`asin(x)`	Returns the arc sine of `x`, in radians.
`atan(x)`	Returns the arc tangent of `x`, in radians.
`ceil(x)`	Returns the smallest integer that is greater than or equal to `x`.
`cos(x)`	Returns the cosine of `x` in radians.
`degrees(x)`	Assuming `x` is an angle in radians, the function returns the angle converted to degrees.
`exp(x)`	Returns e^x
`floor(x)`	Returns the largest integer that is less than or equal to `x`.
`hypot(x, y)`	Returns the length of a hypotenuse that extends from (0, 0) to (`x`, `y`).
`log(x)`	Returns the natural logarithm of `x`.
`log10(x)`	Returns the base-10 logarithm of `x`.
`radians(x)`	Assuming `x` is an angle in degrees, the function returns the angle converted to radians.
`sin(x)`	Returns the sine of `x` in radians.
`sqrt(x)`	Returns the square root of `x`.
`tan(x)`	Returns the tangent of `x` in radians.

The `math` Module (cont'd.)

The math module defines variables pi and e, which are assigned the mathematical values for *pi* and *e*

- Can be used in equations that require these values, to get more accurate results

Variables must also be called using the dot notation

- Example:
  - circle_area = math.pi * radius**2

slido

Please download and install the Slido app on all computers you use

What does the following code do?

import math

ⓘ Start presenting to display the poll results on this slide.

Basic Import

To import an entire module, simply use the import statement followed by the module name. You can then access the module's functions, classes, and variables using dot notation.

```
import math
result = math.sqrt(16)
print(result)  # Output: 4.0
```

Import Specific Functions or Variables

from module_name import attribute1, attribute2, ...

- The from ... import statement in Python is used to import specific attributes (such as functions, classes, or variables) from a module or package into the current namespace.
- This allows you to use these attributes directly without prefixing them with the module name, making the code more concise and readable.

slido

Please download and install the Slido app on all computers you use

Which of the following is a correct way to import only the sqrt function from the math module?

ⓘ Start presenting to display the poll results on this slide.

Example:

```
from math import sqrt

result = sqrt(16)
print(result)  # Output: 4.0
```

This statement imports only the sqrt function from the math module, allowing you to use sqrt directly without prefixing it with math.

result = sqrt(16):This calls the sqrt function directly, computing the square root of 16.

print(result):This prints the result, which is 4.0.