Fairfield UNIVERSITY

CPSC 1101
Introduction to Computing

School of Engineering & Computing
Dept. of Computer Science & Engineering
Dr. Sidike Paheding

Agenda

- Recap
- Lecture
- In-class exercises
- Reading assignment: **pages 72 – 83**

Fairfield UNIVERSITY

Recap on Python Operators

Plickers Card Game

Recap Summary

- If you use multiple operators in one expression, you can use parentheses to clarify the sequence of operations. Otherwise, Python applies its order of precedence.
- You can use the logical operators to join two or more Boolean expressions. This can be referred to as a *compound conditional expression*.
- The AND and OR operators only evaluate the second expression if necessary. As a result, they are known as short-circuit operators.

Fairfield UNIVERSITY

What if we have a *condition* for either logical or relational operators?

Conditional Statements

If…

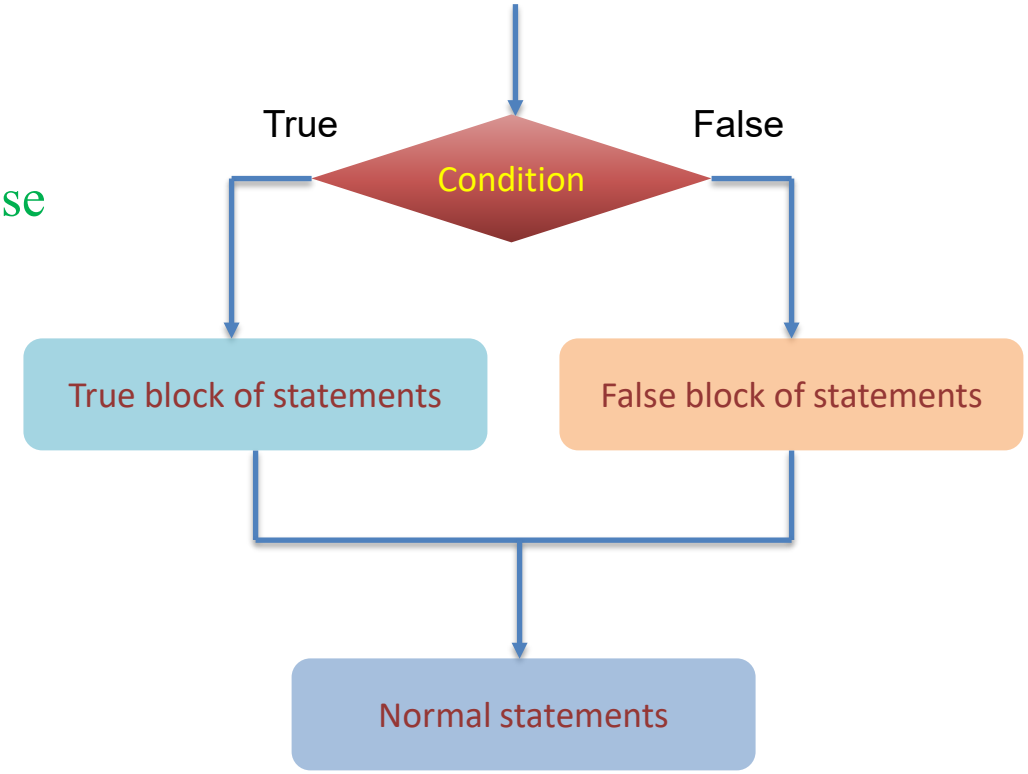
else…

Selection Structure

- All languages provide statements that implement the *selection structure*.
- In Python, this structure is implemented by one type of statement: the *if* statement.
- An *if statement* lets you control the execution of statements based on the results of a *Boolean expression*.
- To code an if clause, you code the *if* keyword followed by a *Boolean expression* and a **colon**. Then, you code a block of one or more **indented** statements starting on the next line. That block ends when the indentation ends.

Selection Statement (aka Conditional Statement)

```
if BOOLEAN EXPRESSION (CONDITION):
    # if condition evaluates to True
    True block of statements
else:
    # if condition evaluates to False
    False block of statements
```



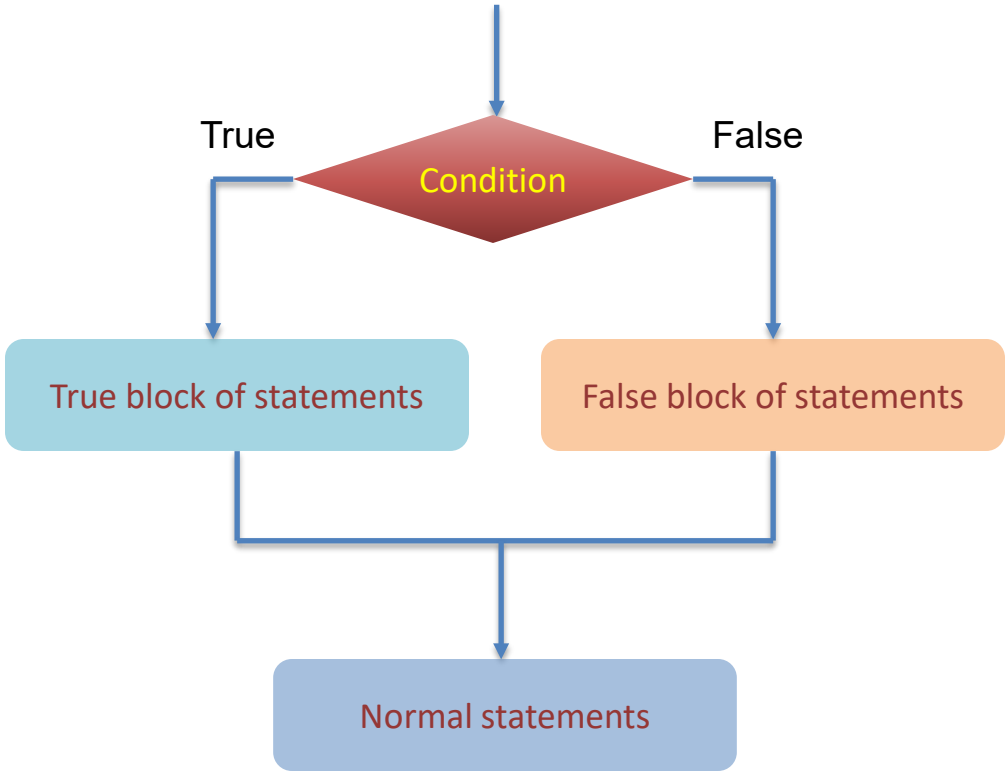
Examples

Example 1 (*else* omitted)

```
x = 10
if x < 0:
    print("x,",  x, " is negative.")
print("This is always printed")
```

Example 2

```
x = 10
if x < 0:
   print("x,",  x, " is negative.")
else:
    print("x,",  x, " is positive.")
print("This is always printed")
```



Examples: *Nested Conditionals*

```
x = 10
y = 10

if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

Python uses indentation to indicate a block of code.

Indentation!

If not

"IndentationError: expected an indented block after 'if' statement on line 2"

Exercise 1

Write a Python script that determines whether a user's input is odd or even.

The syntax of the multiple *elif* statement

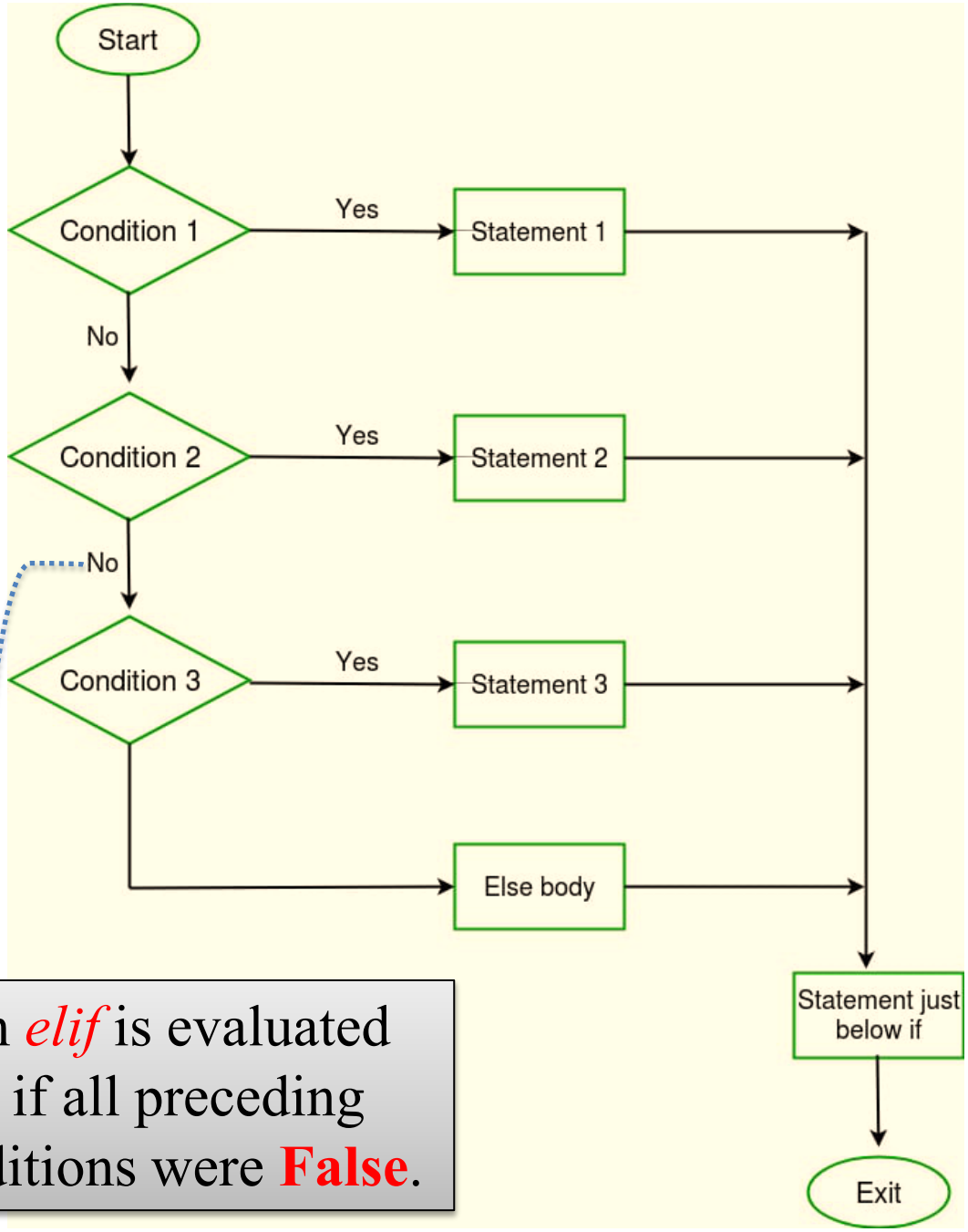
The elif (short for "else if") statement adds additional conditions to an *if statement*, enabling you to check multiple expressions sequentially.

```
if BOOLEAN EXPRESSION 1 (CONDITION 1):
        STATEMENTS_1        # if condition evaluates to True
elif BOOLEAN EXPRESSION 2 (CONDITION 2):
        STATEMENTS_2        # if condition evaluates to True
elif BOOLEAN EXPRESSION 3 (CONDITION 3):
        STATEMENTS_3        # if condition evaluates to True
…
…
else:
        STATEMENTS_n        # if none of the above conditions evaluates to True
```

Each *elif* is evaluated only if all preceding conditions were **False**.

Multiple if and *elif* statement

```
if Condition 1:
    Statement 1
elif Condition 2:
    Statement 2
elif  Condition 3:
    Statement 3
else:
    Else body
Statement just below if
```



Each *elif* is evaluated only if all preceding conditions were **False**.

Example: *Chained Conditions*

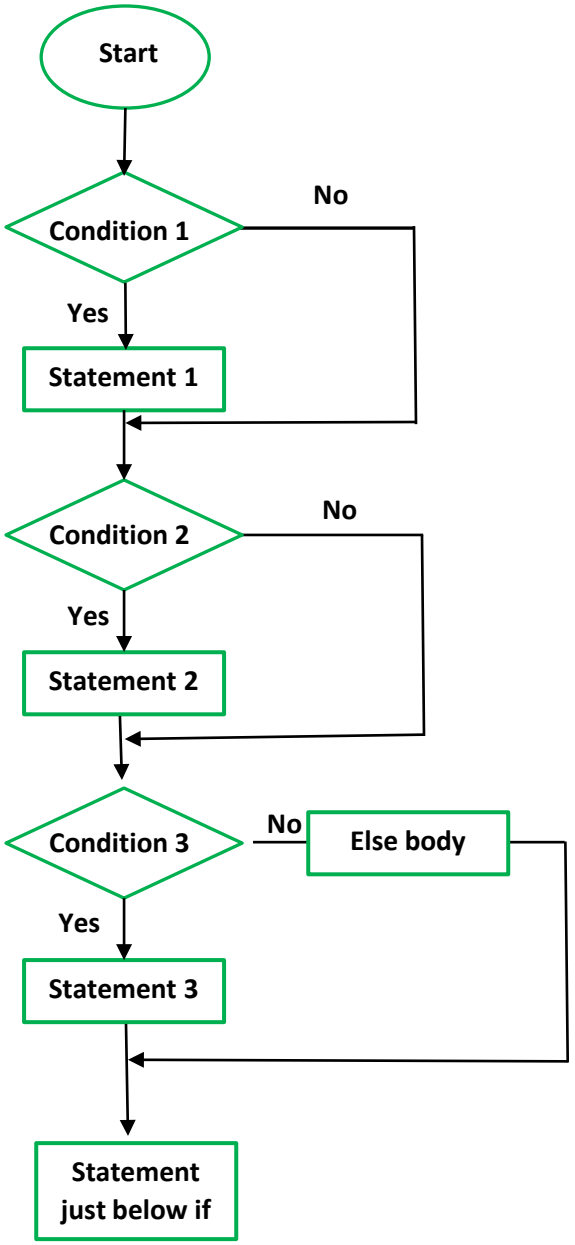
```
x=4
if x < 0:
    print("The negative number ",  x, " is not valid here.")
elif x > 0:
    print(x, " is a positive number")
else:
    print(x, " is 0")
```

Indentation!

Multiple if statement

In Python, multiple if statements can be used to evaluate several **conditions independently.**

```
if Condition 1:
    Statement 1
if Condition 2:
    Statement 2
if  Condition 3:
    Statement 3
else:
    Else body
Statement just below if
```



Example: Multiple if statement

```
user_permissions = ['read', 'write', 'execute']

if 'read' in user_permissions:
    print("User can read files.")

if 'write' in user_permissions:
    print("User can write files.")

if 'delete' in user_permissions:
    print("User can delete files.")
```

```
User can read files.
User can write files.
```

if Statements vs. if...elif...else

Feature	Multiple if Statements	if...elif...else Statements
Evaluation of Conditions	All conditions are evaluated.	Conditions are evaluated sequentially until one is True.
Execution of Code Blocks	All code blocks with True conditions are executed.	Only the code block of the first True condition is executed.
Use Case	When multiple conditions may be True and need to be handled.	When only one condition can be True, and you need to choose one action.

Exercise 2

Write a Python script that asks the user to enter a number between 1 and 7 and prints the corresponding day of the week (i.e. 3 => Tuesday)

Exercise 3

Calculating worker's pay:

A worker is paid according to his hourly wage up to 40 hours, and 50% more for overtime. Write a Python script that calculates the pay to a worker. The script asks the user to enter the number of hours and the hourly wage, and then displays the pay.

Exercise 4

Write a Python script that accepts a student's numerical grade in a course and displays the letter grade according the table below.

Numerical Grade	Letter Grade
Greater than or equal to 90	A
Greater than or equal to 80 and less than 90	B
Greater than or equal to 70 and less than 80	C
Greater than or equal to 60 and less than 70	D
Less than 61	F

nested *if statements*

- In many programs, you will need to code one *if statement* within a clause of another *if statement* to get the logic right. The result is known as *nested if statements*.
- They allow you to check for further conditions only if the previous condition(s) are met.
- This enables more granular control over the flow of your program by creating multiple layers of decision-making.

nested *if statements*

```
if condition1:
    # Code block executed when condition1 is True
    if condition2:
        # Code block executed when condition1 and condition2 are True
        if condition3:
            # Code block executed when condition1, condition2, and condition3 are True
            ...
        else:
            # Code block executed when condition1 and condition2 are True, but condition3 is False
    else:
        # Code block executed when condition1 is True, but condition2 is False
else:
    # Code block executed when condition1 is False
```

nested *if statements*

```
username_input = "user123"
password_input = "pass123"

stored_username = "user123"
stored_password = "pass123"
account_locked = False

if username_input == stored_username:
    if not account_locked:
        if password_input == stored_password:
            print("Login successful.")
        else:
            print("Incorrect password.")
    else:
        print("Account is locked.")
else:
    print("Username not found.")
```

nested *if – elif* statements

```
if condition1:
    # Code block executed when condition1 is True
    if condition2:
        # Code block executed when condition1 and condition2 are True
    elif condition3:
        # Code block executed when condition1 is True and condition2 is False but condition3 True
    else:
        # Code block executed when condition1 is True and both condition2 and condition3 are False
elif condition4:
    # Code block executed when condition1 is False but condition4 is True
    if condition5:
        # Code block executed when condition1 is False, condition4 is True, and condition5 is True
    else:
        # Code block executed when condition1 is False, condition4 is True, and condition5 False
else:
    # Code block executed when condition1 and condition4 are False
    if condition6:
        # Code block executed when condition1 and condition4 are False, but condition6 is True
    elif condition7:
        # Code block executed when condition1 and condition4 are False, condition6 is False, but
condition7 is True
    else:
        # Code block executed when none of the above conditions are True
```

nested *if statements*

- In the example below, the outer if statement checks for two customer type codes.
- The if clause tests for the "r" code, and the elif clause tests for the "w" code.
- If neither test evaluates to True, the else clause sets the discount_percent variable to zero.

Type code	Invoice total	Discount percent
R (for Retail)	< 100	0
	>= 100 and < 250	.1
	>= 250	.2
W (for Wholesale)	< 500	.4
	>= 500	.5

Nested if statements for applying customer discounts

```
customer_type = "R"
invoice_total = 150
if customer_type.lower () == "r":
    if invoice_total < 100 :
        discount_percent = 0
    elif invoice_total>= 100 and invoice_total< 250 :
        discount_percent = .1
    elif invoice_total>= 250 :
        discount_percent = .2
elif customer_type.lower () == "w":
    if invoice_total < 500 :
        discount_percent = .4
    elif invoice_total>= 500 :
        discount_percent = .5
else:
    discount_percent = 0
print(discount_percent)
```

Type code	Invoice total	Discount percent
R (for Retail)	< 100	0
	>= 100 and < 250	.1
	>= 250	.2
W (for Wholesale)	< 500	.4
	>= 500	.5

Summary - nested *if statements*

- Nested if Statements are used to make decisions that depend on previous decisions.
- **Control Flow**: The inner if statements are only evaluated if their parent if statements are True.
- **Use Cases**: Useful in scenarios where conditions are hierarchical and dependent on one another.
- **Best Practices**: Keep nesting to a minimum, consider alternative approaches, and maintain code readability.

pseudocode to plan if statements

- When if statements get complicated, though, you 1nay want to do some planning before you start coding. For that, we recommend the use of *pseudocode*.
- Pseudocode is just your personal shorthand for what needs to be done.
- Within the pseudocode, though, you can use the sa1ne coding structures that are required by Python.
- If you compare the Python code to the pseudocode, you can see how they relate.

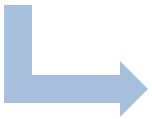
Pseudocode for planning the processing of test score entries

```
Get test score
IF score is from 0 to 100
    add score to total score
    add 1 to the number of scores
ELSE IF score = 999
    print end of program message
ELSE
    print error message
```

pseudocode

Pseudocode for planning the processing of test score entries

Get test score
IF score is from 0 to 100
    add score to total score
    add 1 to the number of scores
ELSE IF score = 999
    print end of program message
ELSE
    print error message



The Python code that's based on the pseudocode

```
total_score = 0
score_counter = 0
score = int(input("Enter test score: "))
if score >= 0 and score <= 100:
    total_score += score
    score_counter += 1
elif score == 999:
    print("Ending program...")
else:
    print("Test score must be from 0 through 100. Score discarded.")
```

Tips for Effective Use if-statements

- Limit Nesting Depth:
  - Aim to keep nesting to two or three levels when possible.
- Use Comments Wisely:
  - Document the purpose of each condition, especially in complex nested structures.
- Refactor When Necessary:
  - If the nested logic becomes too complex, consider refactoring using functions or other control structures.
- Consistent Indentation:
  - Use consistent indentation levels to enhance readability.
- Test Thoroughly:
  - Ensure all possible paths through the nested conditions are tested.

Exercise 5

You are designing a survival game where a player must make decisions based on their health, inventory, and the current time of day. The following is the rule:

- If the player's health is below 30:
  - If it's daytime, the player should "Rest"
  - If it's nighttime, the player should "Rest" regardless of other factors.
- If the player's health is 30 or above:
  - If the player has food in their inventory:
    - If it's daytime, the player should "Explore."
    - If it's nighttime, the player should "Eat."
  - If the player does **not** have food in their inventory:
    - If it's daytime, the player should "Search for Food. "
    - If it's nighttime, the player should "Rest."

Exercise 5

```
``` Python pseudocode
SET health to player's current health
SET has_food to True if the player has food in inventory, otherwise False
SET time_of_day to either "daytime" or "nighttime"

IF health < 30:
    SET action to "Rest"
ELSE:
    IF has_food is True:
        IF time_of_day is "daytime":
            SET action to "Explore"
        ELSE:  # time_of_day is "nighttime"
            SET action to "Eat"
    ELSE:  # has_food is False
        IF time_of_day is "daytime":
            SET action to "Search for Food"
        ELSE:  # time_of_day is "nighttime"
            SET action to "Rest"

PRINT action
```
```

Fairfield UNIVERSITY

CPSC 1101
Introduction to Computing

School of Engineering & Computing
Dept. of Computer Science & Engineering
Dr. Sidike Paheding