Conditional Statements

Dr. Dongchul Kim

Statement

- A statement is the fundamental building block of a programming language, serving as the smallest standalone element that directs the execution flow of a program. These statements come in various forms, each with a specific purpose. For instance:
    - Assignment Statements: These are used to assign values to variables, like `x = 3`.
    - Conditional Statements: These statements, like if-else blocks, guide the program to execute different code paths based on certain conditions.
    - Loop Statements: These are used for iterating over a sequence of values or executing a block of code repeatedly.
    - Method Call Statements: These involve invoking functions or methods to perform specific tasks.
- And there are many other types of statements, each playing a crucial role in building the logic and functionality of a program.
-

Expression

- An entity in programming refers to a single element that can represent a value or a combination of values. These entities can be constants, variables, or expressions formed by combining these elements. Some examples include:
    - Individual Numbers or Constants: Like 5, which represents a fixed value.
    - Variables: Such as x, which can hold various values during the execution of a program.
    - Arithmetic Combinations: Expressions like a + b, which combine two entities through an arithmetic operation.
    - Comparison Expressions: Such as x <= y or x == y, which compare values and yield a Boolean result.
    - Logical Expressions: Like x and y, which perform logical operations on entities.
- These entities are fundamental in creating and manipulating data within a program, allowing for complex calculations, decision-making, and logical operations.

Conditional Statements

`if` statement

The if statement is a control structure in programming used to execute a specific block of code only if a certain condition is met. It evaluates a Boolean expression, which results in either True or False. The syntax of an if statement typically looks like this:

```
if boolean_expression:

    # Statements to execute if the boolean_expression is True

    statement1

    statement2

    # ... more statements if needed
```

Here, boolean_expression represents a condition that evaluates to a Boolean value. If this condition is True, the **indented block of code (the body of the if statement)** is executed. If boolean_expression is False, the code inside the if block is skipped. This allows for conditional execution of code based on the evaluation of the expression.

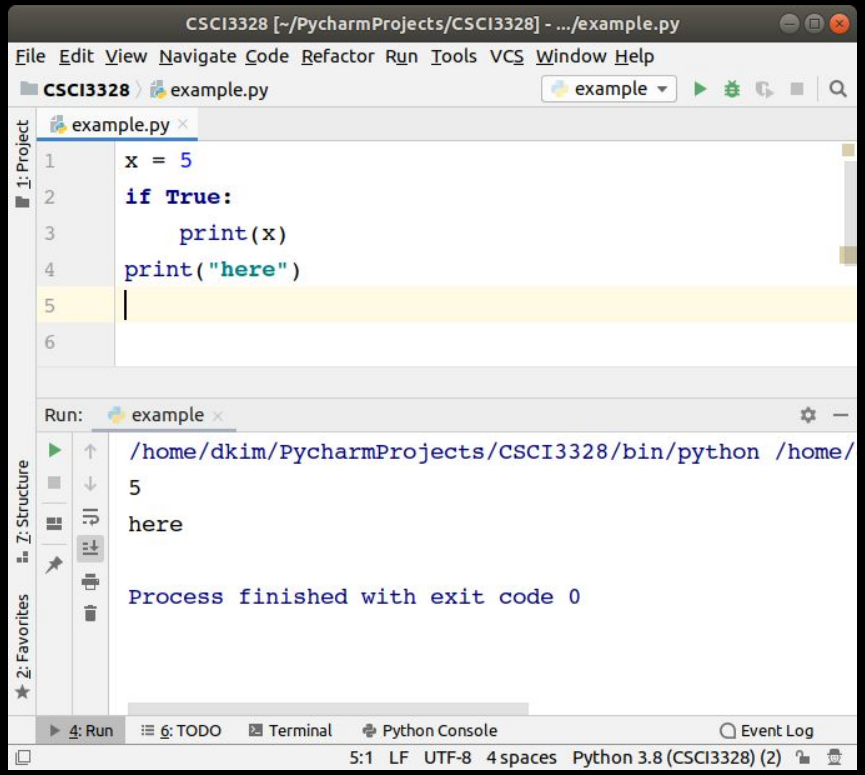
`if` statement head and body

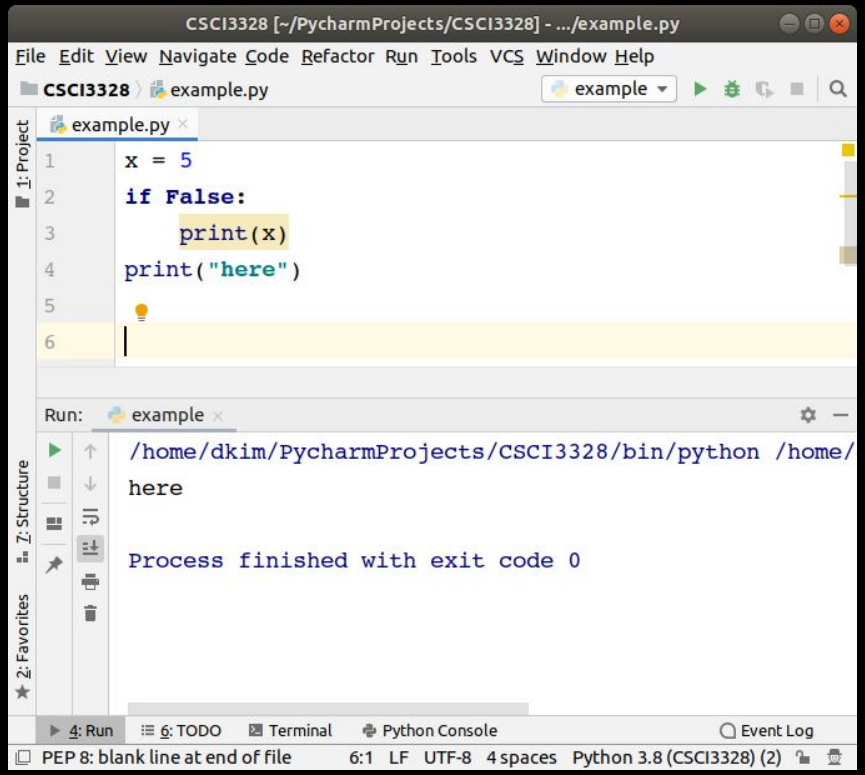
```
if x > 5: # if statement head

    print("x is" , x)        # statement body

    print(x, "is greater 5") # statement body

print("We print this even if x is less than 5")
```

The body of an if statement is defined by indentation, which can be achieved using either a tab or a consistent number of spaces (commonly four spaces).

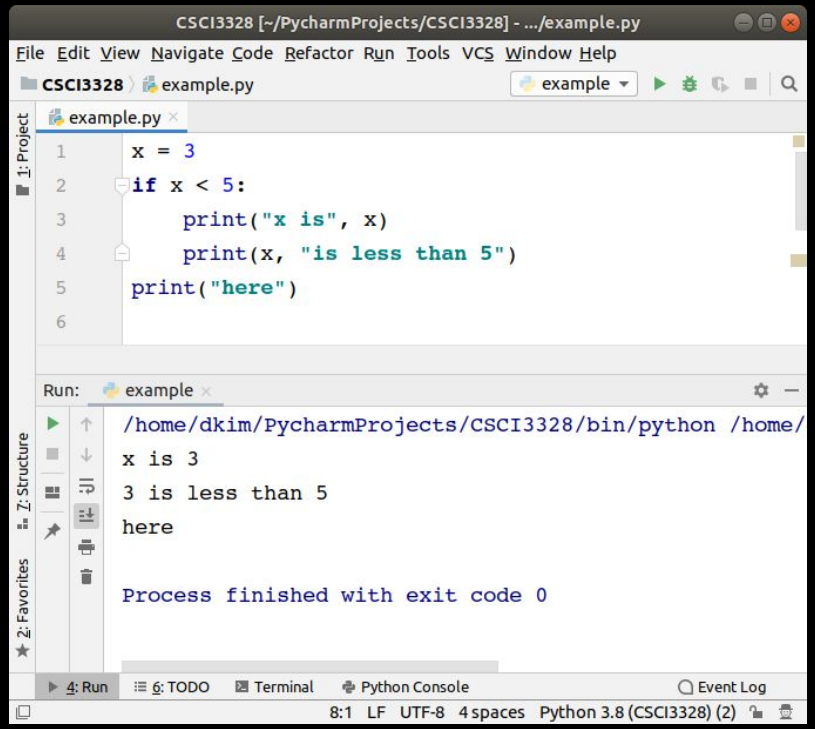
This indentation distinguishes the statements that should be executed when the if statement's condition is True.

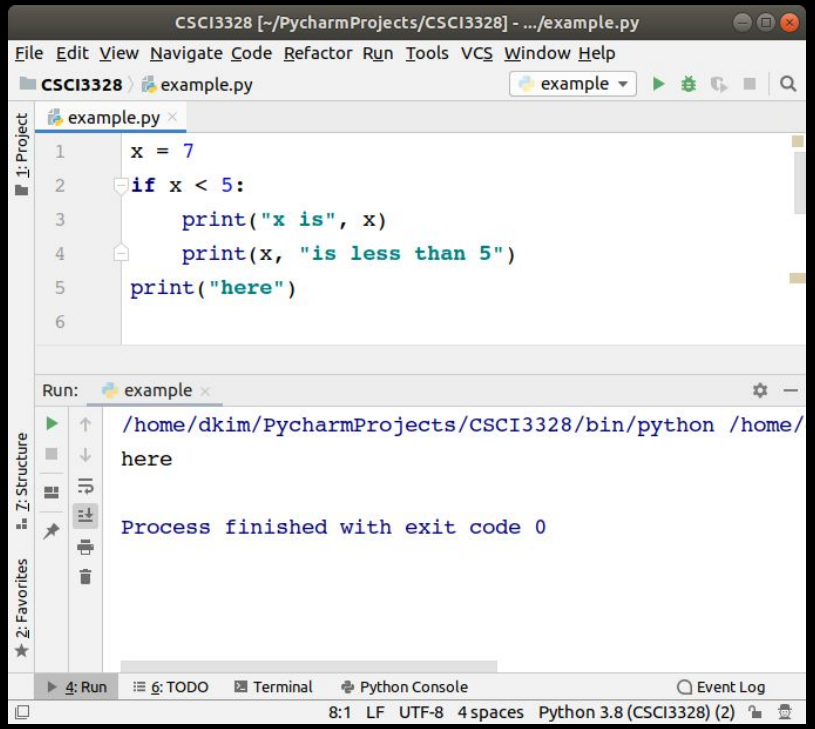
Examples


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = 5
2 if True:
3 print(x)
4 print("here")
5
6
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/
5
here
Process finished with exit code 0
4: Run 6: TODO Terminal Python Console Event Log
5:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = 5
2 if False:
3 print(x)
4 print("here")
5
6
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/
here
Process finished with exit code 0
4: Run 6: TODO Terminal Python Console Event Log
PEP 8: blank line at end of file 6:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

Examples


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = 3
2 if x < 5:
3 print("x is", x)
4 print(x, "is less than 5")
5 print("here")
6
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/
x is 3
3 is less than 5
here
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
8:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = 7
2 if x < 5:
3 print("x is", x)
4 print(x, "is less than 5")
5 print("here")
6
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/
here
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
8:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

`else` statement

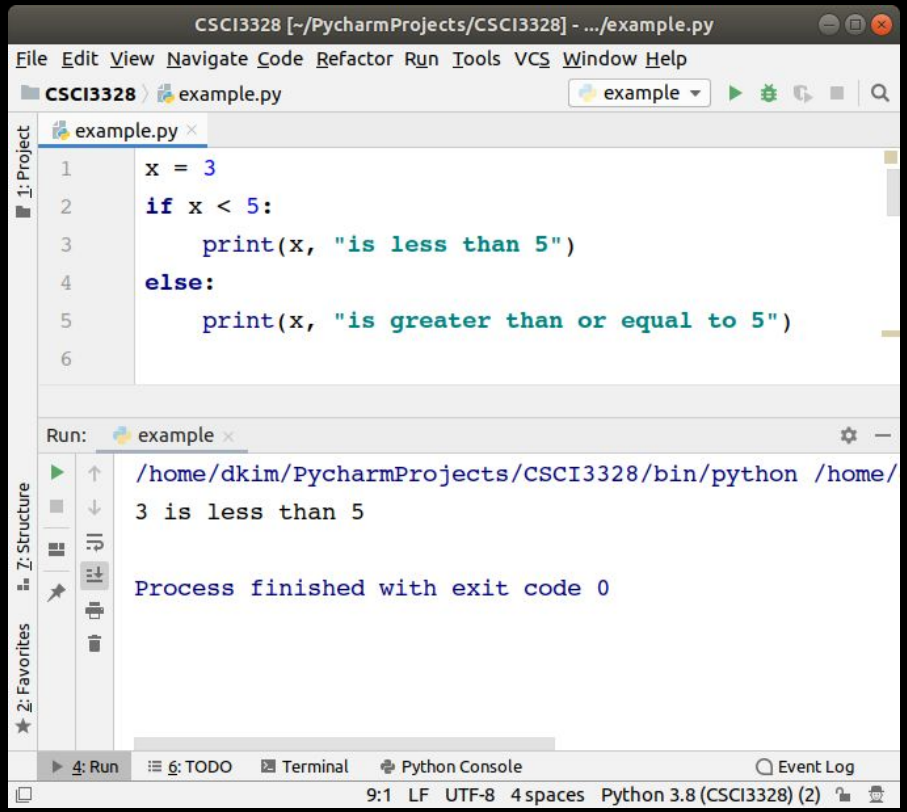
The `else` keyword is used with `if` together.

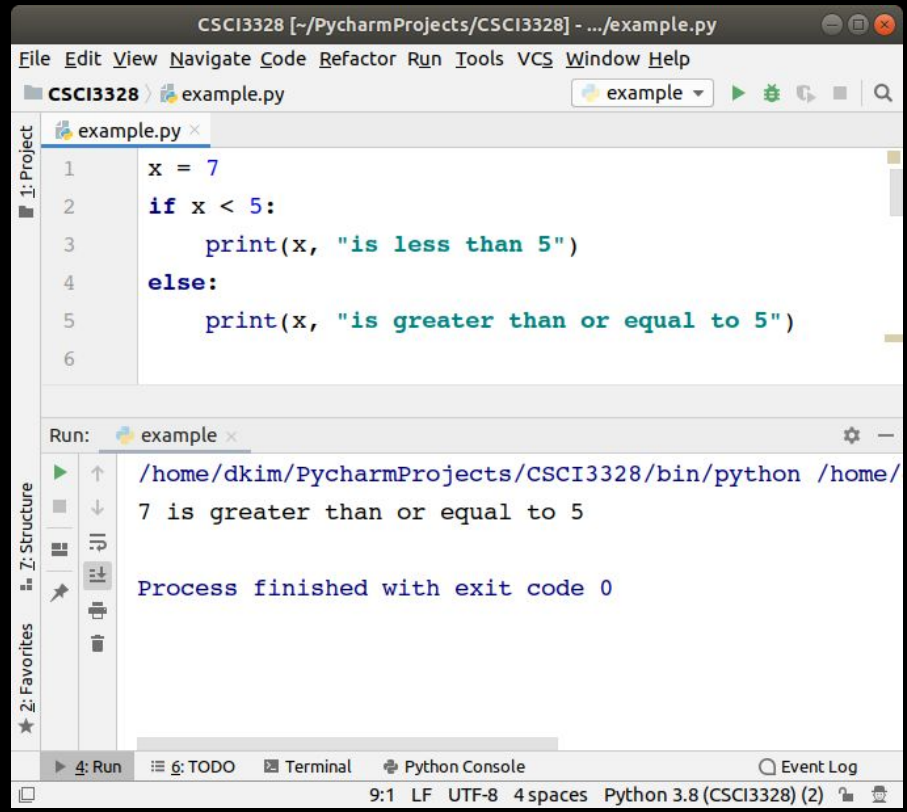
`else` should be located after `if` statement.

`else` catches anything which isn't caught by the preceding `if` conditions.

```
if boolean_expression:
    statements
else:
    statements # executed when boolean_expression is False
```

Examples


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = 3
2 if x < 5:
3 print(x, "is less than 5")
4 else:
5 print(x, "is greater than or equal to 5")
6
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/
3 is less than 5
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
9:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = 7
2 if x < 5:
3 print(x, "is less than 5")
4 else:
5 print(x, "is greater than or equal to 5")
6
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/
7 is greater than or equal to 5
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
9:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

Lab 8-1

Get a number (int) from the user using an input method and assign the value into a variable. If the value is an odd number, display a string, the number is odd. Otherwise, display it is even. The output to be printed on the console should follow the format as follows. Blue indicates the value user inputs.

```
Please input a number? 17(enter)
Processing……
17 is an odd number.
```

Lab 8-1

Submission:

Capture the output of your program.

Upload both the captured image files and Python files on Blackboard.

Lab 8-1

Number Input (20 Points):
Correctly prompting for a number input and assigning it to a variable (20 Points)
Incorrect or missing number prompt (0 Points)

Processing Display (10 Points):
Displaying "Processing……" after receiving the input (10 Points)
Missing or incorrect processing message (0 Points)

Odd or Even Determination (40 Points):
Accurately determining and displaying if the number is odd or even (40 Points)
Incorrect determination or display format (0 Points)

Correct Output Format (30 Points):
Displaying the output in the format "[input number] is an odd/even number." (30 Points)
Incorrect or missing output format (0 Points)

Total Points: 100

`elif` statement

`elif` stands for `else if` of other programming languages. `elif` statement is similar to `if` statement but it should be located between `if` statement and `else` statement.

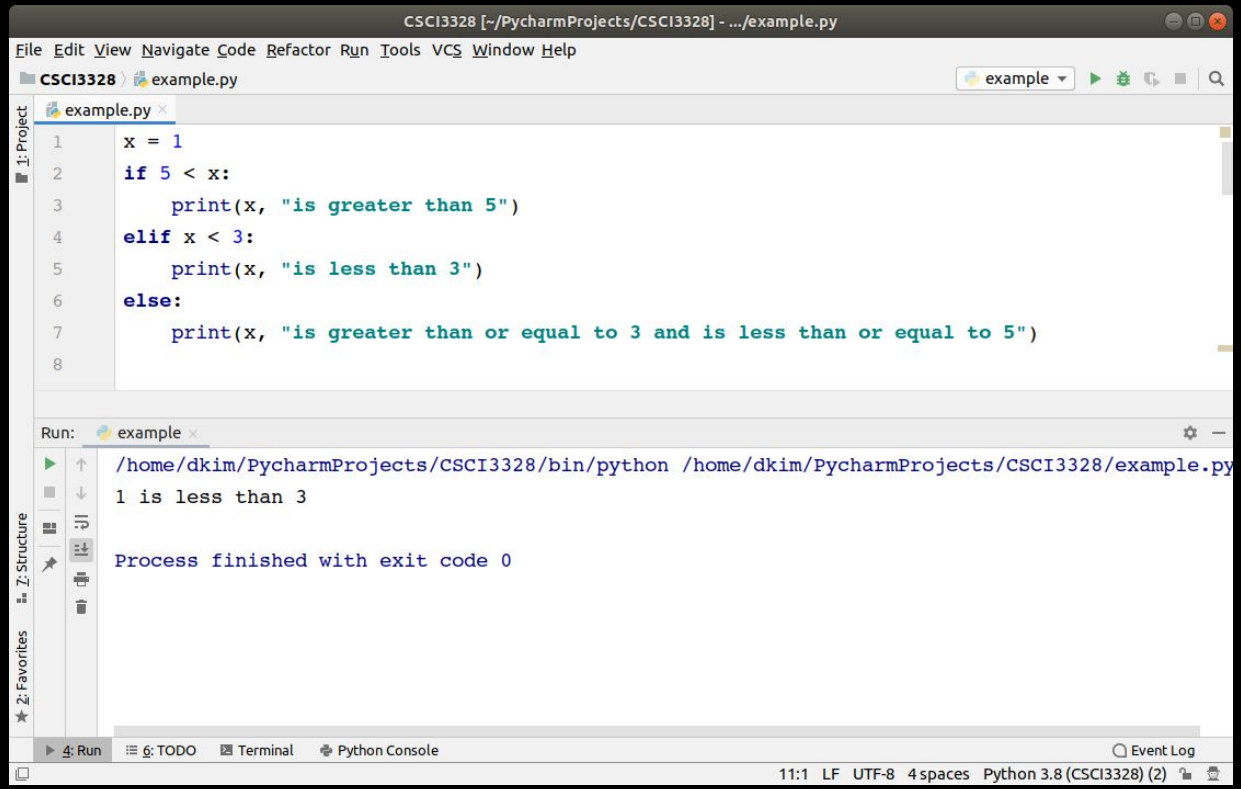
If the boolean expression of the `if` statement is `False`, the next `elif` statement (if it exists) is executed.

If the boolean expression of `elif` statement is `True`, the body of the `elif` statement will be executed, and then rest of `elif` and `else` are ignored (skipped).

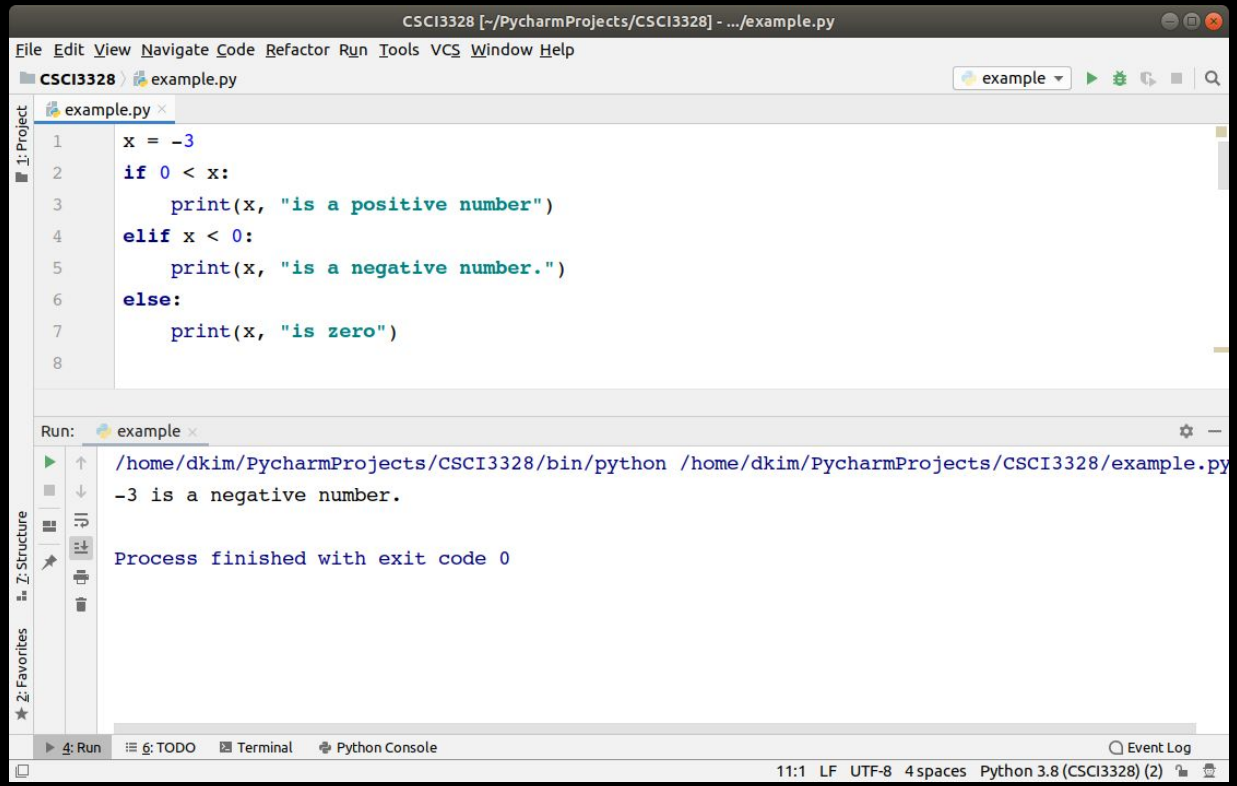
`elif` statements

```
if boolean_expression:
    statements
elif boolean_expression:
    Statements
elif boolean_expression: # you can put as many elif as needed
    statements
else:
    statements
```

Example


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = 1
2 if 5 < x:
3 print(x, "is greater than 5")
4 elif x < 3:
5 print(x, "is less than 3")
6 else:
7 print(x, "is greater than or equal to 3 and is less than or equal to 5")
8
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
1 is less than 3
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
11:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

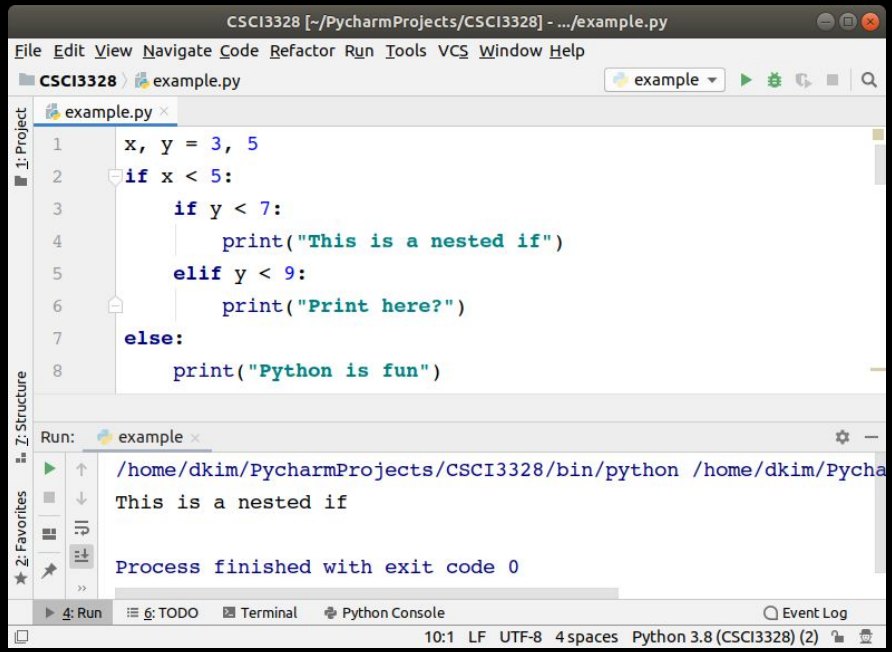
Example


CSCI3328 [~/PycharmProjects/CSCI3328] - .../example.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CSCI3328 example.py
example
example.py
1 x = -3
2 if 0 < x:
3 print(x, "is a positive number")
4 elif x < 0:
5 print(x, "is a negative number.")
6 else:
7 print(x, "is zero")
8
Run: example
/home/dkim/PycharmProjects/CSCI3328/bin/python /home/dkim/PycharmProjects/CSCI3328/example.py
-3 is a negative number.
Process finished with exit code 0
1: Project
7: Structure
2: Favorites
4: Run
6: TODO
Terminal
Python Console
Event Log
11:1 LF UTF-8 4 spaces Python 3.8 (CSCI3328) (2)

Nested `if` statements

If an `if` statement appears inside another `if` statement, it is called a nested `if` statement.

The nested `if` is executed only if the outer `if` statement results in a `True` condition.



Lab 8-2 ~ 8-10

- Submission
  - Capture the output of your program.
  - Upload both the captured image files and Python files on Blackboard.
  - Create a separate file for each lab; do NOT combine multiple labs into a single file.

Lab 8-2

This assignment aims to develop your understanding of basic Python programming concepts such as user input, conditional statements, and arithmetic operations. You will write a program to calculate the Grade Point Average (GPA) based on letter grades for three courses.

```
Enter your grades (e.g., A B C): A B C
Your GPA is: 3.0
```

Lab 8-2

Input Letter Grades:
Prompt the user to input letter grades for three courses in one line, separated by spaces. For instance, the user might input: A B C.
Use input().split() to read and split the input into individual grades.

Convert Letter Grades to Points:
Assign numerical values to each letter grade as follows: A = 4.0, B = 3.0, C = 2.0, D = 1.0, F = 0.0.
Implement if and elif statements to convert each letter grade to its corresponding numerical value.

Calculate GPA:
Calculate the total score by adding the numerical values of the three grades.
Compute the GPA by dividing the total score by 3. Use the formula: gpa = total_score / 3.
Print the calculated GPA.

Submission:
Save your Python script as a .py file.
Submit the file to the designated assignment section on your learning platform.

Lab 8-2

User Input for Grades (20 Points):
Correctly prompting for and capturing letter grades for three courses (20 Points)
Incorrect or missing prompt for grades (0 Points)

Grade Conversion to Numeric Values (30 Points):
Accurately converting letter grades to corresponding numeric values for GPA calculation (30 Points)
Incorrect or incomplete grade conversion (0 Points)

GPA Calculation (30 Points):
Correctly calculating the GPA based on the numeric values of the letter grades (30 Points)
Incorrect GPA calculation (0 Points)

Correct Output Format (20 Points):
Displaying the calculated GPA in the format "Your GPA is: [calculated GPA]" (20 Points)
Incorrect or missing output format (0 Points)

Total Points: 100

Lab 8-3

Get a number (int, float) from the user using an input method and assign the value into a variable. If the value is an positive number, display a string, "the number is positive." Otherwise, "negative" or "zero". The output to be printed on the console should follow the format as follows. Blue indicates the value user inputs.

```
Please input a number? 17(enter)
Processing……
17 is a positive number.
```

Lab 8-3

Number Input (20 Points):
Correctly prompting for a number input (int or float) and assigning it to a variable (20 Points)
Incorrect or missing number prompt (0 Points)

Processing Display (10 Points):
Displaying "Processing……" after receiving the input (10 Points)
Missing or incorrect processing message (0 Points)

Number Classification (50 Points):
Accurately classifying and displaying if the number is positive, negative, or zero (50 Points)
Incorrect classification or display format (0 Points)

Correct Output Format (20 Points):
Displaying the output in the format "[input number] is a positive/negative/zero number." (20 Points)
Incorrect or missing output format (0 Points)

Total Points: 100

Lab 8-4

This assignment is designed to enhance your understanding and application of conditional statements in Python. You will write a program to determine the maximum and minimum values among three given integer numbers without using built-in functions for maximum and minimum.

```
Enter three integer numbers: 12 7 19
Maximum number is: 19
Minimum number is: 7
```

Lab 8-4

Input Three Integer Numbers:

Prompt the user to input three separate integer numbers. Use input() to receive each number.

Determine Maximum and Minimum:

Use if, elif, and else statements to compare the three numbers.
Identify the maximum number by comparing each number with the others.
Similarly, identify the minimum number.

Print the Results:

Display the maximum and minimum numbers.

Lab 8-4

User Input for Three Integers (20 Points):
Correctly prompting for and capturing three integer numbers (20 Points)
Incorrect or missing prompt for integers (0 Points)

Implementation of Conditional Statements (40 Points):
Accurately using conditional statements to determine the maximum and minimum values among the three integers without using built-in functions (40 Points)
Inaccurate or incorrect use of conditional statements (0 Points)

Correct Output for Maximum Value (20 Points):
Displaying the maximum number among the three integers in the format "Maximum number is: [maximum number]" (20 Points)
Incorrect or missing maximum value output (0 Points)

Correct Output for Minimum Value (20 Points):
Displaying the minimum number among the three integers in the format "Minimum number is: [minimum number]" (20 Points)
Incorrect or missing minimum value output (0 Points)

Total Points: 100

Lab 8-5

- Make a program that determines if given lengths of three line segments can make a triangle or not.
- Input format

  `L1  L2  L3`
- Output format

  `Yes/No`

```
Please input lengths of three line segments:  3 4 5(enter)
Yes
```

```
Please input lengths of three line segments:  3 14 5(enter)
No
```

Lab 8-5

Input for Line Segment Lengths (20 Points):
Correctly prompting for and capturing the lengths of three line segments (L1, L2, L3) (20 Points)
Incorrect or missing prompt for line segment lengths (0 Points)

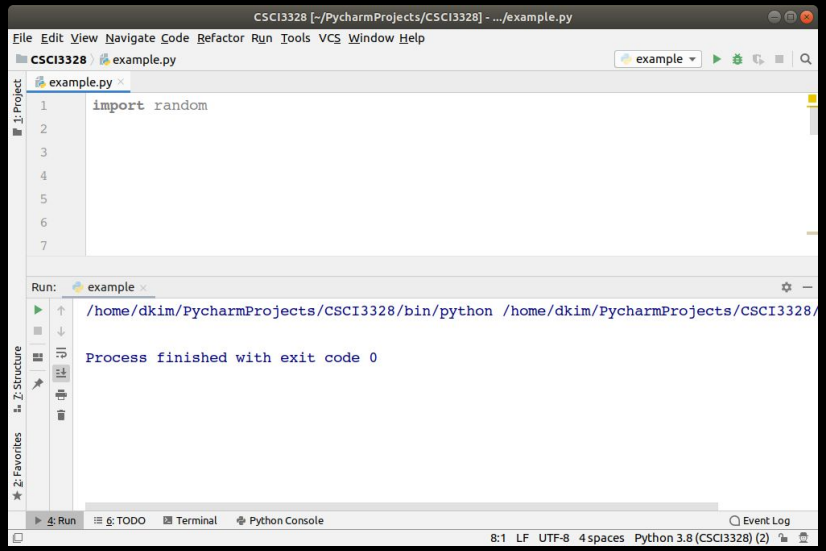
Triangle Feasibility Logic (50 Points):
Accurately applying the triangle inequality theorem to determine if the given lengths can form a triangle (50 Points)
Incorrect application of the triangle feasibility logic (0 Points)

Correct Output Format (30 Points):
Displaying "Yes" if the lengths can form a triangle, or "No" if they cannot, in the specified output format (30 Points)
Incorrect or missing output format (0 Points)

Total Points: 100

`random` module

To generate a random number in Python, you can utilize the random module. To access the functionalities of this module, it needs to be imported at the beginning of your program.



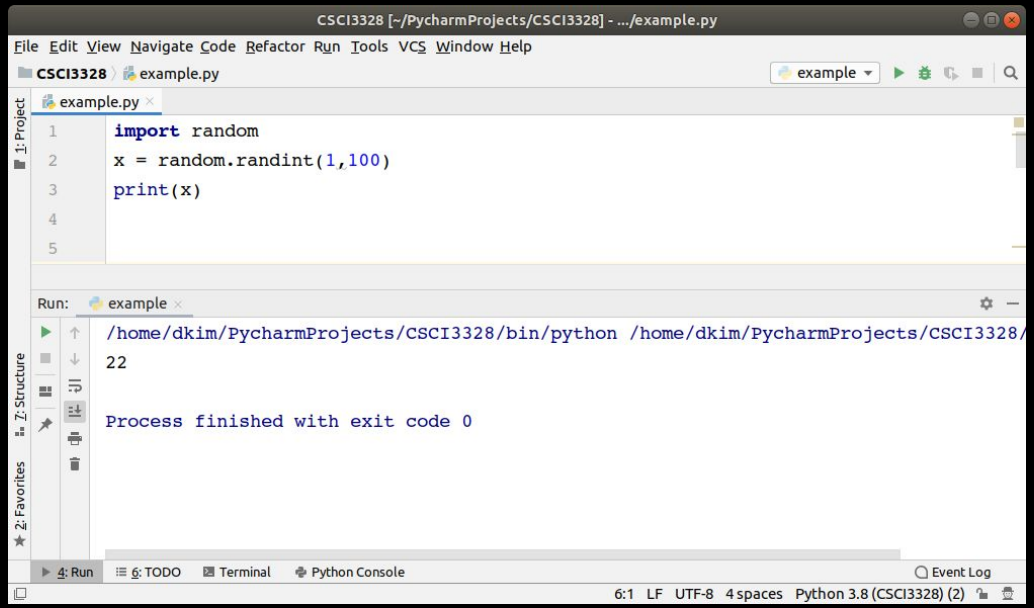
`random.seed()` and `random.random()`

Python's random module generates numbers that are pseudo-random, meaning they are determined by an initial value known as a seed. Therefore, these numbers are not truly random but appear random for most practical purposes. The random.random() function within this module produces a floating-point number between 0 and 1.

```
import random

# Set a seed value for reproducibility

seed_value = 42

random.seed(seed_value)

# Generate a random float between 0 and 1 using the seeded random number generator

random_number = random.random()

# Print the generated random number

print("Random Number with Seed:", random_number)
```

Random integer

The random package includes the randint(start, end) function, which is used to produce a random integer within the specified range, inclusive of both the start and end values. This function returns the generated random integer.



Lab 8-6

Generate a random integer number between 1 and 3 using `random` package and `randint()` function. (i.e. 1, 2, or 3)

```
Random number generation
2
```

Lab 8-6

Importing Random Package (20 Points):
Correctly importing the random package in the Python script (20 Points)
Failure to import or incorrect import statement (0 Points)

Using randint() Function (40 Points):
Accurately using the randint() function from the random package to generate a random integer between 1 and 3 (40 Points)
Incorrect use of the randint() function or range error (0 Points)

Display of Generated Number (40 Points):
Correctly displaying the randomly generated number (1, 2, or 3) (40 Points)
Incorrect or missing display of the random number (0 Points)

Total Points: 100

Lab 8-7

You are given four integers `a`, `b`, `c` and `d`. Determine if there's a rectangle such that the lengths of its sides are `a`, `b`, `c` and `d` (in any order).

Input: integers `a`, `b`, `c` and `d`.

Output: Print a single line containing one string "`YES`" or "`NO`".

For example,

```
Input:  1 1 2 2
Output: YES
```

```
Input:  3 2 2 3
Output: YES
```

```
Input:  1 2 2 2
Output: NO
```

Lab 8-7

Input Capture (20 Points):
Correctly capturing four integer inputs a, b, c, and d (20 Points)
Incorrect or missing input capture (0 Points)

Rectangle Feasibility Logic (60 Points):
Accurately applying logic to determine if the four integers can form the sides of a rectangle (i.e., checking if there are two pairs of equal lengths) (60 Points)
Incorrect or faulty rectangle feasibility logic (0 Points)

Correct Output Format (20 Points):
Displaying "YES" if the integers can form a rectangle, or "NO" if they cannot, in the specified output format (20 Points)
Incorrect or missing output format (0 Points)

Total Points: 100

Lab 8-8

Develop a rock-paper-scissors Game. Implement that the computer randomly throws using random number generation (1, 2, or 3)

The rock-paper-scissors game rule is here. https://en.wikipedia.org/wiki/Rock_paper_scissors

```
Start Game
1. Rock
2. Paper
3. Scissors
What do you want to throw? 3(enter)

Computer:Paper vs You:Scissor

You win!!!
```

Lab 8-8

Game Initialization and User Input (20 Points):
Successfully displaying game start prompt and options (Rock, Paper, Scissors) (10 Points)
Correctly capturing the user's choice (1, 2, or 3) (10 Points)
Incorrect or missing game initialization/user input (0 Points)

Random Computer Choice Generation (30 Points):
Correctly implementing random number generation for computer's choice (1 for Rock, 2 for Paper, 3 for Scissors) (30 Points)
Incorrect or faulty random generation (0 Points)

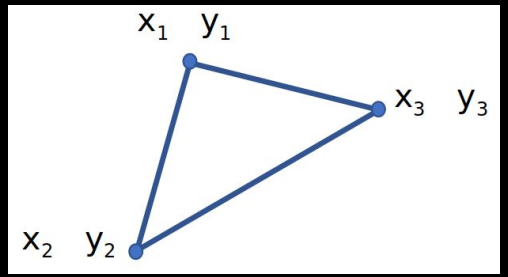
Game Logic and Result Display (30 Points):
Accurately implementing the game logic to determine the winner (30 Points)
Incorrect or faulty game logic (0 Points)

Outcome Presentation (20 Points):
Clearly displaying the choices (Computer: [Choice] vs You: [Choice]) (10 Points)
Appropriately announcing the game result ("You win!", "You lose!" or "It's a tie!") (10 Points)
Incorrect or unclear outcome presentation (0 Points)

Total Points: 100

Lab 8-9

Make a program that returns the area of a triangle given three points of the triangle as parameters, and then call the function with test values (6 arguments, x_1, y_1, x_2, y_2, x_3, y_3) that the user inputs like below.




```
Please input three points :  2 1 8 9 1 8 (enter)
The area is 25
```

Lab 8-9 Hint

Calculates the area and boundary length of a triangle with three points.

https://www.cuemath.com/geometry/area-of-triangle-in-coordinate-geometry/

Lab 8-9

User Input for Triangle Points (20 Points):
Correctly prompting for and capturing six input values representing the coordinates of the triangle's three points (x1, y1, x2, y2, x3, y3) (20 Points)
Incorrect or missing prompt for points (0 Points)

Function Implementation (30 Points):
Accurately implementing a function to calculate the area of a triangle given its three points (30 Points)
Incorrect function implementation or calculation method (0 Points)

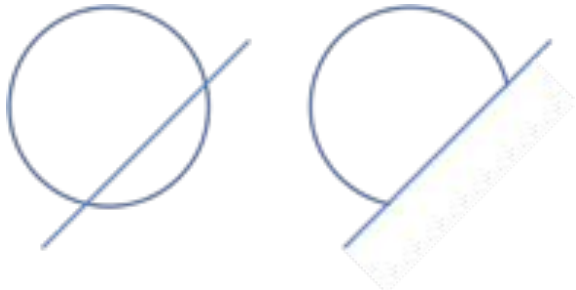
Correctness of Area Calculation (30 Points):
Correctly calculating the area of the triangle using the provided points (30 Points)
Incorrect area calculation (0 Points)

Output Display (20 Points):
Displaying the calculated area in the format "The area is [calculated area]" (20 Points)
Incorrect or missing output format (0 Points)

Total Points: 100

Lab 8-10

Create a program that computes the area of a geometric shape formed by slicing a circle with a straight line. The program should first determine if the line effectively divides the circle and then calculate the area of the larger resulting shape.



Input:

```
x  y // Origin of circle
r   // Radius of circle
x1  y1  x2  y2 // two points on a line
```

Output:

```
a   // Area of the shape after cutting
```

Lab 8-10

Example

Input:

```
4   4

2

8   6   3   1
```

Output:

```
11.42
```

Lab 8-10 Hint

Find all angles of a given triangle

https://www.geeksforgeeks.org/find-angles-given-triangle/

Area of the circle sector segment

https://www.mathsisfun.com/geometry/circle-sector-segment.html

Find intersections of a circle and line

https://mathworld.wolfram.com/Circle-LineIntersection.html

Lab 8-10

Circle and Line Input Capture (10 Points):
Accurately prompting for and capturing the circle's origin (x, y) and radius (r) (5 Points)
Correctly prompting for and capturing the coordinates of two points on a line (x1, y1, x2, y2) (5 Points)
Incorrect or missing input for circle or line (0 Points)

Line and Circle Intersection Validation (20 Points):
Effectively determining if the line intersects the circle and creates a slice (20 Points)
Incorrect or failed validation of intersection (0 Points)

Area Calculation of Larger Segment (50 Points):
Accurately calculating the area of the larger segment after slicing the circle (50 Points)
Incorrect area calculation (0 Points)

Correct Output Display (20 Points):
Displaying the calculated area in the format "a: [calculated area]" (20 Points)
Incorrect or missing output format (0 Points)

Total Points: 100