CS65: Introduction to Computer Science

Complex Boolean Expressions
While Loop

Md Alimoor Reza
Assistant Professor of Computer Science

What to work on

Lab #3 has been released today. It will be due on October 4th for all sections

Text Reading

https://python.swaroopch.com/control_flow.html

"The While Statement" in Control Flow section

Review: Exercise #1

Prompt the user to enter a string.

when user enters “Red” then print(“I like red channel of an RGB image.”)
when user enters “Green” then print(“I like green channel of an RGB image.”)
when user enters “Blue” then print(“I like blue channel of an RGB image.”)
when user enters anything else then print(“Image format is not correct.”)

Use ONLY if-elif-else statements to print out an appropriate response based on the inputted number and the above conditions

Review: Exercise #2

Prompt the user for the number of hours of sleep they got last night.

Use ONLY if-elif-else statements to print out an appropriate response based on the inputted number and the following chart

Above 8	"You are well-rested!"
Between 4 and 8	"The coffee shop is around the corner."
Between 0 and 4	"Are you sure you are awake?"
Less than 0	"input error."

Review: Example - Consider the problem

Prompt the user to enter an integer number. Then it determines the following:

5, 15, etc	**Odd and divisible by 5**
7, 21, etc	**Odd and divisible by 7**
19, 57, etc	**Odd and divisible by 19**

Use ONLY if-elif-else statements to print out an appropriate response based on the inputted number and the above conditions

5, 10, 15, etc	**Odd and divisible by 5**
7, 14, 21, etc	**Odd and divisible by 7**
19, 38, 57, etc	**Odd and divisible by 19**

Review Solution:

example_if_elif_else.py

```
1  # alimoor reza
2  # 09/23/2025
3
4  number = int(input("Enter an integer number: "))
5
6  if number%2 == 1 and number%5 == 0:
7      print(number, " is odd and divisible by 5")
8  elif number%2 == 1 and number%7 == 0:
9      print(number, " is odd and divisible by 7")
10 elif number%2 == 1 and number%19 == 0:
11     print(number, " is odd and divisible by 19")
12 else:
13     print(number, " doesn't satisfy our criteria")
```

Review: Exercise #3

Prompt the user to enter the age of a person

Infant	From birth up to about 1 year old
Toddler	If the age is between 1 to 3 years old
Child	If the age is between 4 to 12 years old
Teenager	If the age is between 13 to 19 years old
Adult	If the age is between 20 to 39 years old
Middle-aged	If the age is between 40 to 59 years old
Senior	If the age is 60 and above years old

Use ONLY if-elif-else statements to print out an appropriate response based on the inputted number and the above conditions

Review: Comparison Operators

- We can write expression that can be evaluated to a boolean value with other comparison operators
  - Compare two values or check something

Description	Example	Result
Less than	2 < 15	True
Greater than	2 > 15	False
Less than or equal	2 <= 15	True
Greater than or equal	2 >=15	False
Equality check	2 == 15	False
Inequality check	2 != 15	True

Review: Complex Boolean Expression

- Expressions that are evaluated to two **`bool`** types
- Operations with logical operators — `and/or/not`
  - `and` – given two boolean, are both True? answer is True
    boolean expression$_1$ `and` boolean expression$_2$
  - `or` – given two booleans, at least one is True? answer is True
    boolean expression$_1$ `or` boolean expression$_2$
  - `not` – given a boolean expression, switch between True/False
    `not` boolean expression

Boolean operator	Description
a **and** b	***Boolean AND***: True when both operands are True.
a **or** b	***Boolean OR***: True when at least one operand is True.
not a	***Boolean NOT*** (opposite): True when the single operand is False (and False when operand is True).

Review: Logical Operators

x	y	x and y
False	False	False
False	True	False
True	False	False
True	True	True

- $expression_1$ and $expression_2$

Review: Let's Do More Boolean Expressions

X	Y	X and Y
2 < 15	2 >=15	False
3 < 15	2 ==15	False
3 < 15	15 == 15	True
16 > 15	2 != 15	True

- $expression_1$ and $expression_2$

Examples – and

You can combine Boolean expressions to make more complicated Boolean expressions

and: True when *both* operands are True

```
1  temp = int(input("Enter the temparature outside: "))
2
3  whether = input("Enter the weather type eg, rainy, snowing: ")
4
5  # and operator
6  if temp > 15 and temp < 22:
7      print("It's cold outside")
8
9  if temp <0 and weather == "snowing":
10     print("Check to see if the university is canceled")
```

boolean_expression1.py

Review: Logical Operators

x	y	x or y
False	False	False
False	True	True
True	False	True
True	True	True

$expression_1$ or $expression_2$

Examples – or

You can combine Boolean expressions to make more complicated Boolean expressions

or: True when *either* of operands are True

```
1  temp = int(input("Enter the temparature outside: "))
2
3  whether = input("Enter the weather type eg, rainy, snowing: ")
4
5  # OR operator
6  if temp > 35 or weather == "sunny":
7      print("Wear sunglass")
8
9  if weather == "rainy" or weather == "snowing":
10     print("bring an umbrella")
```

boolean_expression2.py

Today's content

- More on Complex Boolean Expressions
  - Hands-on Exercise
- While Loops
  - Hands-on Exercise
- Counting Loops
  - Hands-on Exercise
- Infinite Loops
  - Hands-on Exercise

What's wrong with this?

boolean_mistake.py

```
1  answer = input("Are you on campus? yes or no")
2
3  if answer == "yes" or "no":
4      print("I am on campus")
```

boolean_mistake.py

Exercise#1

Download the code, **`BooleanExamples3.py`**

For each if statement, what is the mistake? And how can it be fixed?

boolean_expression3_FIX_ERRORS.py | boolean_expression1.py

```
1  # fix the errors in this code
2  temp = int(input("Enter the temparature outside: "))
3
4  weather = input("Enter the weather type: ")
5
6  if weather == "sunny" and weather == "partly cloudy":
7      print("wear suncreen")
8
9  if weather == "sunny" and temp < 35 or temp > 25 :
10     print("let's have picnic")
11
12 if weather == "sunny" or "partly cloudy":
13     print("let's go to the beach and have fun!")
```

Not Logical Operator

The Python keyword **`not`** will negate any Boolean expression

Turn `True` to `False`

Turn `False` to `True`

x	not x
False	True
True	False

not expression

Introducing not

The Python keyword **`not`** will negate any Boolean expression

Turn `True` to `False`

Turn `False` to `True`

`boolean_expression3.py`

```
1  temp = int(input("Enter the temparature outside: "))
2
3  if not (temp > 15 and temp < 22):
4      print("Don't go outside, it's cold outside!")
```

Any Questions?

And

Or

Not

Repeating Code: While Loops

Sometimes we want a section of our code to repeat multiple times. For this we use a *loop*.

The first loop we're going to look at is a **`while`** loop. Its syntax is a lot like the if statement (but there's no else part):

- Two parts:
  - *Condition* tested for true or false value
  - Statements repeated as long as condition is true

General format:

```
while condition:

  statements
```

What do you think this will do?

```
1  secret_number = 7
2  guess = 0
3
4  while guess != secret_number:
5
6      guess = int(input("Guess a number: "))
7
8      if guess == secret_number:
9          print("That was right, good guess!")
10     else:
11         print("Wrong!")
12
13
14 print("Thanks for playing!")
```

while_loops1.py

Input Validation

input validation: making sure the user's input makes sense for the program

We can use the same pattern as the guess-the-number loop to make an input-validation loop.

Input Validation

input validation: making sure the user's input makes sense for the program

We can use the same pattern as the guess-the-number loop to make an input-validation loop.

```
1  print("Soccer team registration form")
2  num_players = 0
3  while num_players <= 11:
4
5      num_players = int(input("How many players on your team (minimum 11): "))
6
7  print("Registerting team with ", num_players, "players")
```

while_loops2.py

Exercise #2

Prompt the user for a number between 10 and 100

Keep on prompting the user until they enter a valid number

```
give a number between 10 and 100: 112
nope. Try again
give a number between 10 and 100: 2
nope. Try again
give a number between 10 and 100: 56
Thank you!
```

Counting

Loops are good for counting things. A variable that is used to count something is called a **counter**.

Let's say we want to count how many guesses it took for the user to guess the number.

while_loop3.py

```
1  secret_number = 7
2  guess = 0
3  guess_counter = 0
4
5  while guess != secret_number:
6
7      guess = int(input("Guess a number: "))
8      guess_counter = guess_counter + 1
9
10     if guess == secret_number:
11         print("That was right, good guess!")
12     else:
13         print("Wrong!")
14
15 if guess == secret_number:
16     print("Congratulations!")
17 else:
18     print("Sorry!")
```

Recall: +=

Often times, we want to update the value of a variable

```
my_val = my_val + 1
```

Shorthand:

```
my_val += 1
```

Count-controlled loops

When you use a variable to control the number of times the loop executes, it's called a count-controlled loop

```
1  val = 0
2
3  while val < 10:
4      print(val)
5      val += 1 # or val = val + 1
6
7  print("Final val: ", val)
```

while_loops4.py

How will the output of these loops be different?

```
val = 0
while val < 10:
    print(val)
    val += 1
print("all done")
```

```
val = 0
while val < 10:
    val += 1
    print(val)
print("all done")
```

Exercise #3

Write a loop that will print out every integer between 1 and 20

Write a loop that will print out every even integer between 12 and 40

Write a loop that will count down from 10 to 1.

Infinite Loops

In all but rare cases, loops must contain within themselves a way to terminate

Something that makes the test condition false

If your loop does not have a way of stopping, it is called an *infinite loop*

Infinite Loops

In all but rare cases, loops must contain within themselves a way to terminate

Something that makes the test condition false

If your loop does not have a way of stopping, it is called an *infinite loop*

```
counter = 0
while counter < 10:
    print("counter:",counter)
counter += 1
```

Tips for writing loops

- Think about what needs to happen before the loop, during each loop iteration, and after the loop
- Think about what condition should make the loop end
- Make sure there is something that changes inside the loop that will eventually allow the loop's condition to be False
- Set up loop control variables with initial values before the loop

What to work on

Lab #3 has been released today. It will be due on October 4th for all sections

Next Tuesday (09/28), there will be our first Content Quiz on

variable,
expression,
boolean expression
selection statements.

Text Reading

https://python.swaroopch.com/control_flow.html

"The While Statement" in Control Flow section

(Extra) Challenging Exercises

Print out a 2D pattern like this:

XOXOXOXOXOXOXOXOXOXOXOXOXO

OXOXOXOXOXOXOXOXOXOXOXOXOX

XOXOXOXOXOXOXOXOXOXOXOXOXO

OXOXOXOXOXOXOXOXOXOXOXOXOX

Prompt the user for the number of rows and columns they would like.