Python Chapter 3 Decisions Activities

Contents

Time required: 90 minutes

No AI use

Online Tutorials

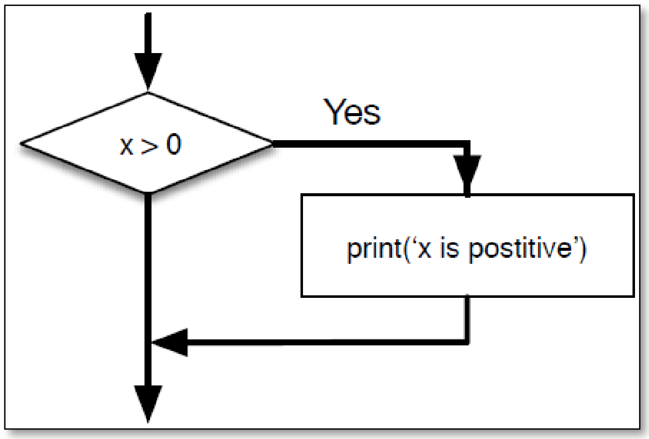
Go through the following tutorials.

- LearnPython.org Conditions

- Python Booleans
- Python If … Else

if

The **if** statement in Python is a control structure used for decision-making. It allows a program to execute specific blocks of code based on whether a given condition evaluates to **True** or **False**.



- **Condition** - The if statement evaluates the condition. This condition can be a comparison, a logical operation, or any expression that results in a boolean value (True or False).
- **Code block -** If the condition is True, the code block immediately following the if statement (indicated by indentation) is executed. If the condition is False, the code block is skipped, and the program continues to the next block of code.

```
age = 19
if age >= 16:
    print("You can drive!")
```

1. **age >= 16** is the boolean expression being evaluated.
2. If the age variable is greater than or equal to 16, the statement prints("You can drive").

Tips

- **Indentation:** Indentation plays a crucial role in Python. The code block that follows the if statement must be indented to indicate that it belongs to that specific if block.

- **Colon** - The colon : at the end of the if statement signifies the beginning of the indented block of code that will execute if the condition is True.
- **Execution** - The code block under if is executed only if the condition is True. If the condition is False, the block is skipped, and the program moves to the next statement after the if block.

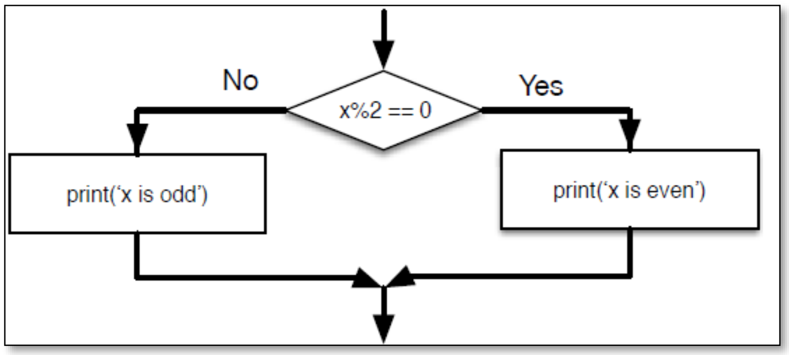
else

The **else** statement is used in conjunction with **if** statements. It provides a way to specify a block of code that should be executed when the condition of the **if** statement evaluates to **`False`**.

The syntax looks like this:

```
if x % 2 == 0:
   print('x is even')
else:
   print('x is odd')
```

If the remainder when x is divided by 2 is 0, then we know that x is even, and the program displays a message to that effect. If the condition is false, the second set of statements is executed.



Since the condition must either be true or false, exactly one of the alternatives will be executed. The alternatives are called branches because they are branches in the flow of execution.

The **else** statement is optional and helps handle situations when the `if` condition is not met. It's a way to define an alternative action when the condition isn't satisfied.

if elif else

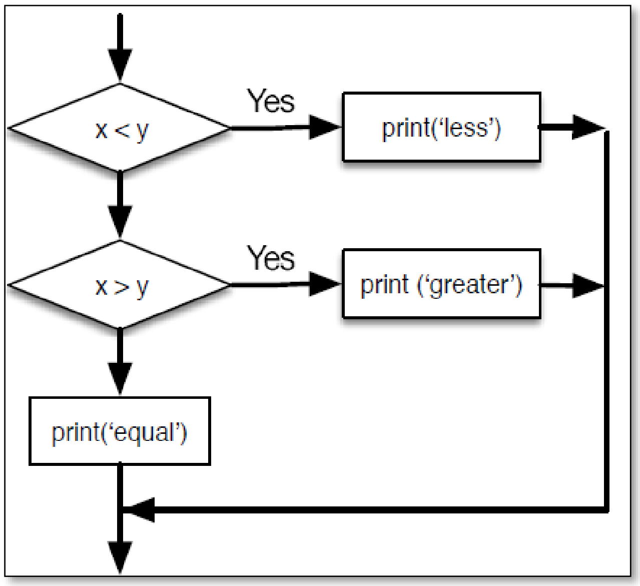
Here are the three keywords that can be used with **if** statements:

- if
- elif
- else

Sometimes there are more than two possibilities, we need more than two branches. One way to express a computation like that is a chained conditional:

```
print("Welcome to the Sunset Bar and Grille.")
age = int(input("Please enter your age: "))
if age >= 21:
    print("Here, have an adult non alcoholic beverage.")
elif age >= 16:
    print("Here have a Coke!")
    print("At least you can drive!")
else:
    print("Here have a Coke!")
print("Thanks for coming to the Sunset Bar and Grille!")
```

elif is an abbreviation of “else if.” Again, exactly one branch will be executed. There is no limit on the number of **elif** statements. If there is an **else** clause, it has to be at the end, but there doesn’t have to be one.



Each condition is checked in order. If the first is false, the next is checked, and so on. If one of them is true, the corresponding branch executes, and the statement ends. Even if more than one condition is true, only the first true branch executes.

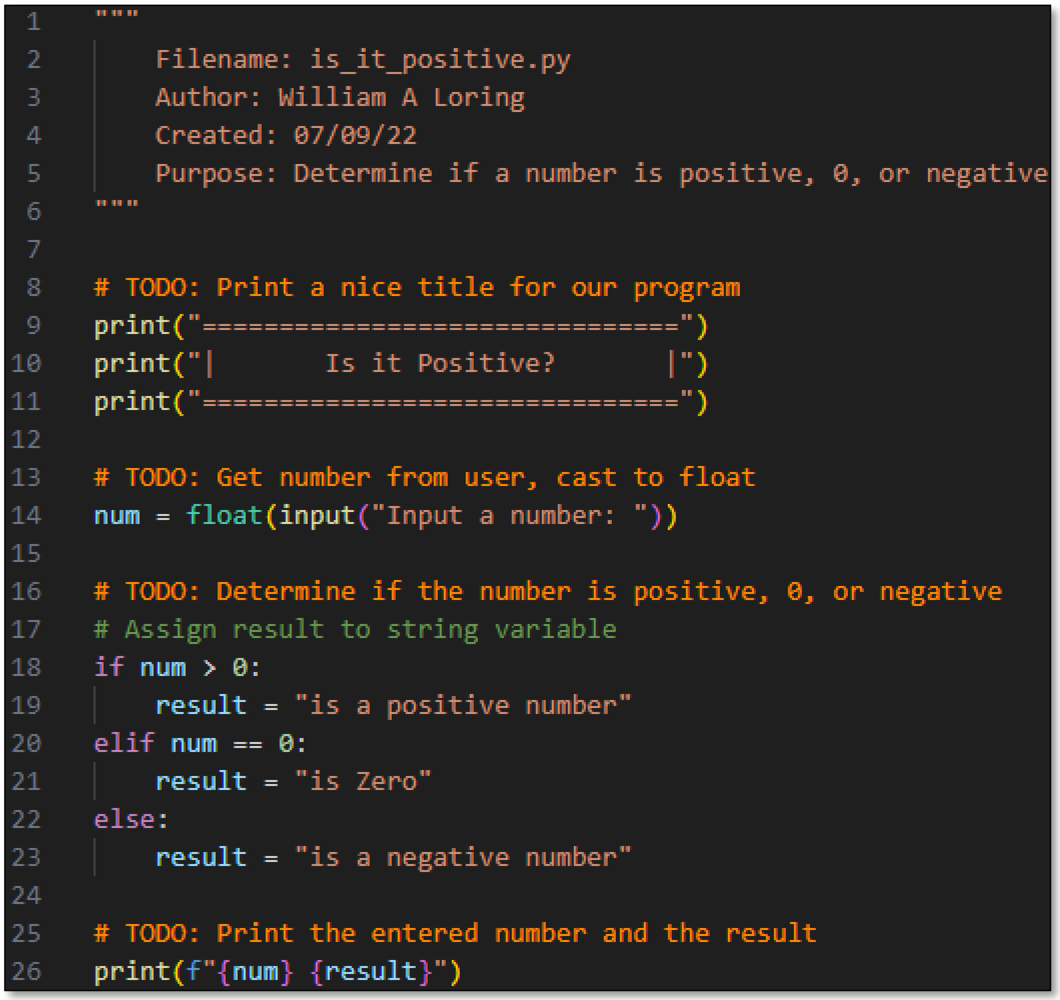
Tutorial 1: Is It Positive?

Let’s have the computer decide if a number is positive, 0, or negative.

1. Create a Python program named **is_it_positive.py**
2. You can pseudocode your code by using a TODO list as shown below. This allows you to plan and start commenting your code.
3. Copy and paste the following to start your program.

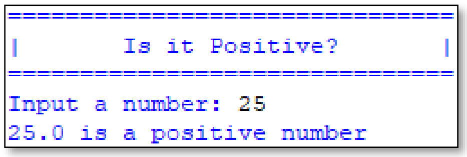
```
"""
    Filename: is_it_positive.py
    Author: William A Loring
    Created: 07/09/22
    Purpose: Determine if a number is positive, 0, or negative
"""

# TODO: Print a nice title for our program

# TODO: Get number from user, cast to float

# TODO: Determine if the number is positive, 0, or negative
# Assign result to string variable

# TODO: Print the entered number and the result
```

4. Enter the following code.

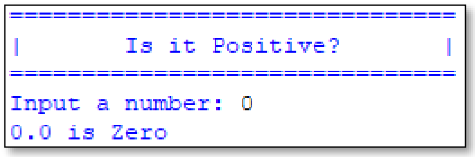


```python
"""
    Filename: is_it_positive.py
    Author: William A Loring
    Created: 07/09/22
    Purpose: Determine if a number is positive, 0, or negative
"""

# TODO: Print a nice title for our program
print("================================")
print("|        Is it Positive?       |")
print("================================")

# TODO: Get number from user, cast to float
num = float(input("Input a number: "))

# TODO: Determine if the number is positive, 0, or negative
# Assign result to string variable
if num > 0:
    result = "is a positive number"
elif num == 0:
    result = "is Zero"
else:
    result = "is a negative number"

# TODO: Print the entered number and the result
print(f"{num} {result}")
```

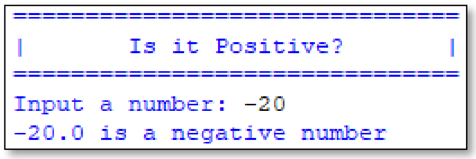
Example runs:



```
================================
|        Is it Positive?       |
================================
Input a number: 25
25.0 is a positive number
```



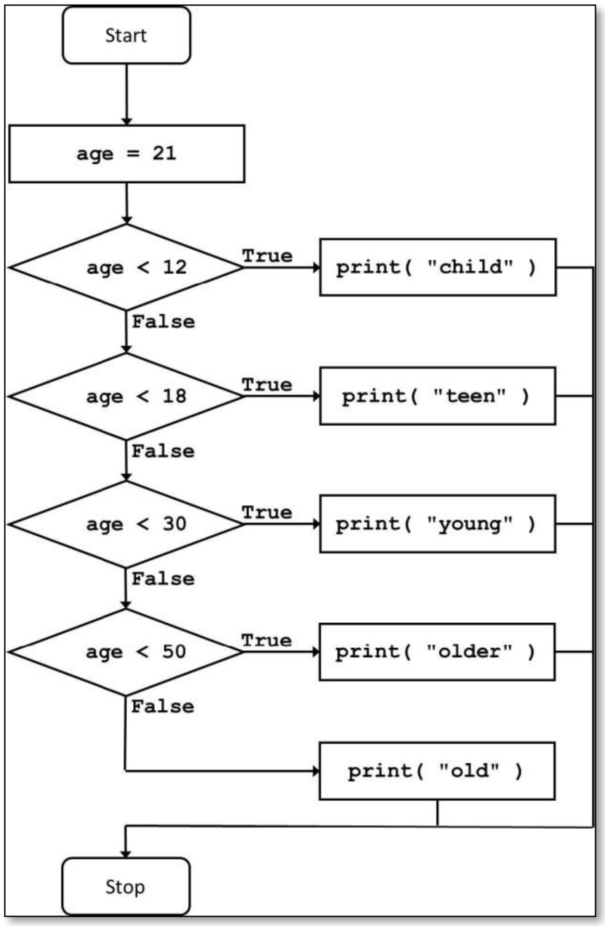
```
================================
|        Is it Positive?       |
================================
Input a number: 0
0.0 is Zero
```



```
================================
|        Is it Positive?       |
================================
Input a number: -20
-20.0 is a negative number
```

Multiple Conditions

When you have multiple conditions, remember what you have eliminated.

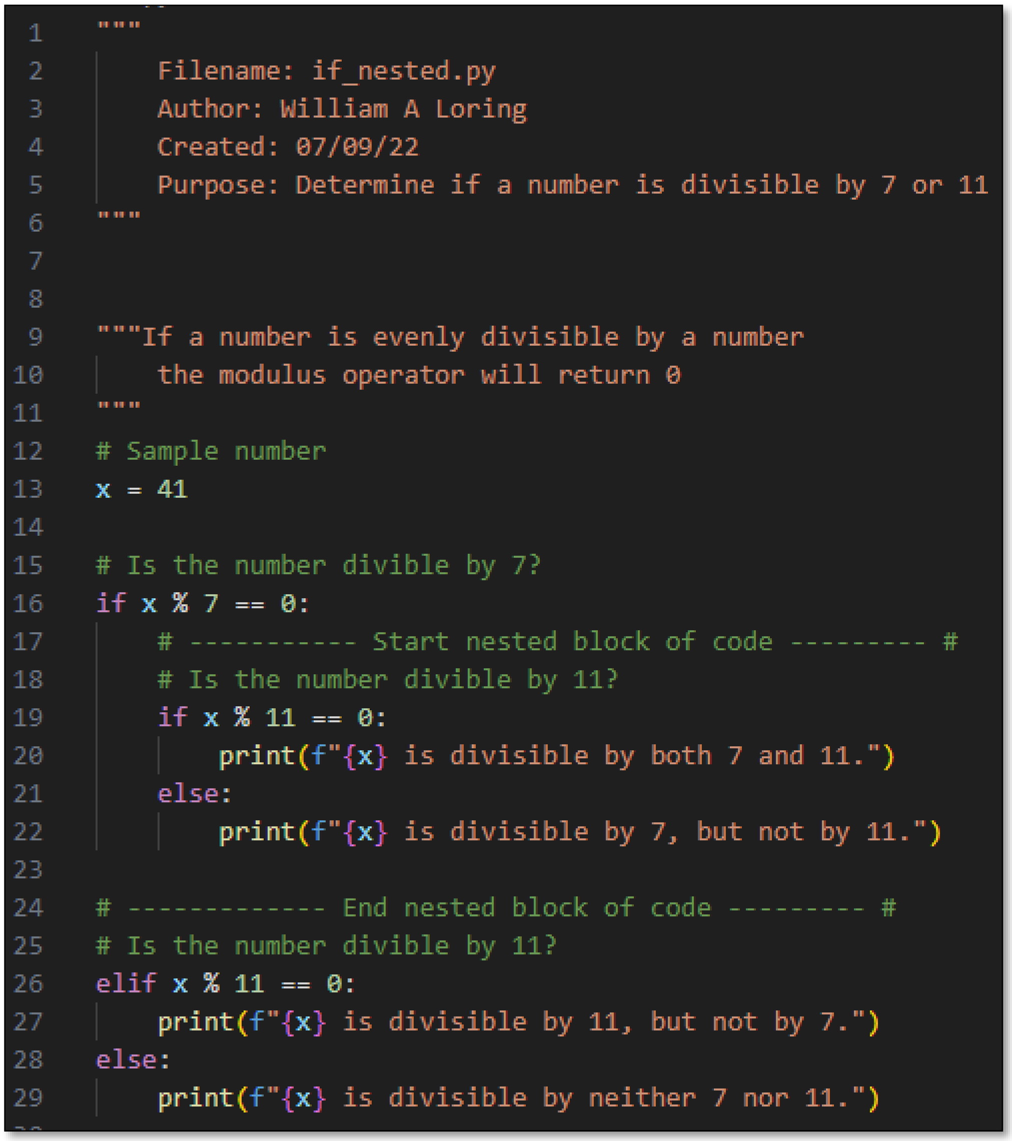


```python
# Extra code and unnecessary conditions
age = 21
if age < 12 and age >= 0:
    print("You're still a child!")
elif age < 18 and age >= 12:
    print("You are a teenager!")
elif age < 30 and age >= 18:
    print("You're pretty young!")
elif age < 50 and age >= 30:
    print("Wisening up, are we?")
else:
    print("Aren't the years weighing heavy?")

# Correct, start at one end or the other and eliminate
age = 21
if age < 12:
    print("You're still a child!")
elif age < 18:
    print("You are a teenager!")
elif age < 30:
    print("You're pretty young!")
elif age < 50:
    print("Wising up, are we?")
else:
    print("Aren't the years weighing heavy?")
```

Tutorial 2: Nesting If Statements

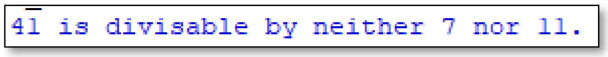
The following tutorial is an example of nested if statements and modulus.

Create Python program named **nesting.py**

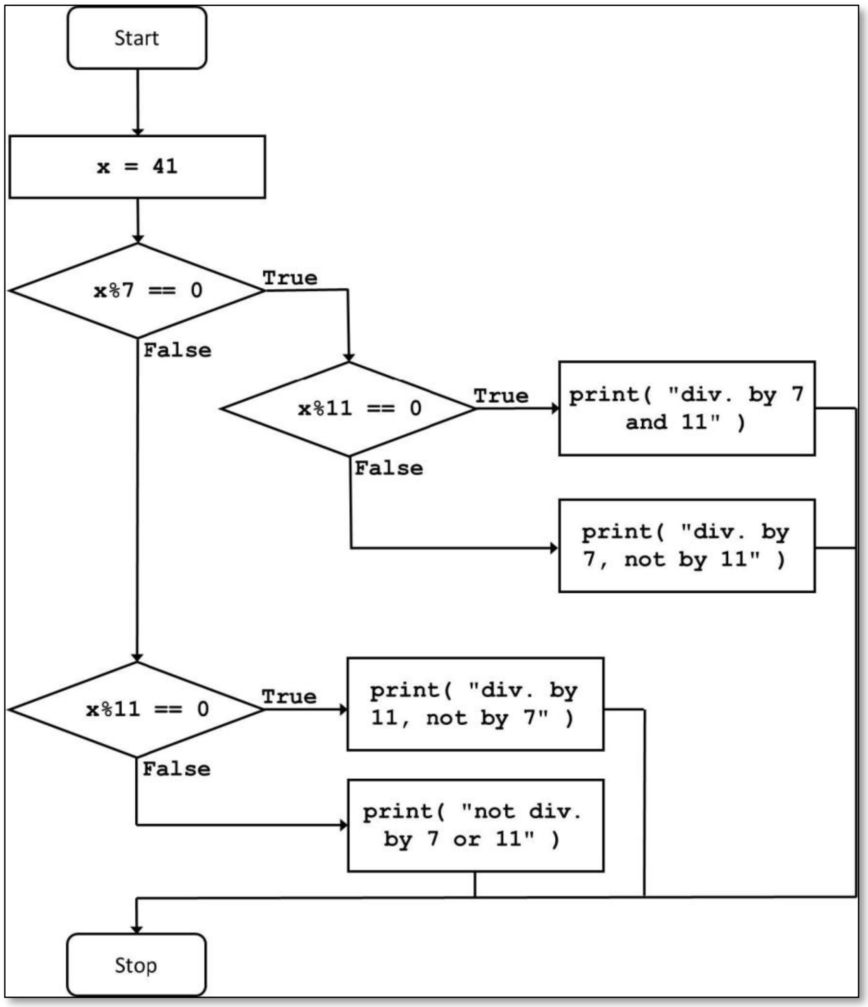


```python
"""
    Filename: if_nested.py
    Author: William A Loring
    Created: 07/09/22
    Purpose: Determine if a number is divisible by 7 or 11
"""


"""If a number is evenly divisible by a number
    the modulus operator will return 0
"""
# Sample number
x = 41

# Is the number divible by 7?
if x % 7 == 0:
    # ----------- Start nested block of code --------- #
    # Is the number divible by 11?
    if x % 11 == 0:
        print(f"{x} is divisible by both 7 and 11.")
    else:
        print(f"{x} is divisible by 7, but not by 11.")

# ------------- End nested block of code --------- #
# Is the number divible by 11?
elif x % 11 == 0:
    print(f"{x} is divisible by 11, but not by 7.")
else:
    print(f"{x} is divisible by neither 7 nor 11.")
```

Example run:



```
41 is divisable by neither 7 nor 11.
```


Start
x = 41
x%7 == 0
True
False
x%11 == 0
True
print("div. by 7 and 11")
False
print("div. by 7, not by 11")
x%11 == 0
True
print("div. by 11, not by 7")
False
print("not div. by 7 or 11")
Stop

Logical Operators

Logical operators in programming, specifically **and** and **or**, are used to combine and evaluate multiple conditions or expressions. They allow you to create more complex conditions by considering the outcomes of multiple smaller conditions.

Logical AND (and)

The **and** operator returns **True** if both conditions it connects are **True.** It returns **`False`** if at least one of the conditions is **False**. For the overall expression connected by **and** to be **True**, all the individual conditions must be **True**.

```
x = 5
y = 10

if x > 0 and y > 0:
    print("Both x and y are positive.")
```

In this example, the message will be printed because both **x** and **y** are greater than 0.

Logical OR (or)

The **or** operator returns **True** if at least one of the conditions it connects is **True**. It returns **False** only if all the conditions are **False**. For the overall expression connected by **or** to be **True**, at least one of the individual conditions must be **True**.

```
age = 15
has_permission = True

if age >= 18 or has_permission:
    print("You can enter the restricted area.")
```

In this example, the **if** statement will be executed and the message will be printed because even though **age** is not greater than 18, the person has permission to enter.

Combining and and or

You can use both **and** and **or** operators in more complex conditions to create a range of possibilities.

```
temperature = 28
time_of_day = "morning"

# The following line uses \ to continue the code on the next line.
if (temperature > 25 and time_of_day == "afternoon") or \
    time_of_day == "morning":
    print("It's a good time for outdoor activities.")
```

In this example, the **if** statement checks whether either the temperature is above 25 degrees Celsius in the afternoon or it's morning. If either of these conditions is met, the message will be printed.

Use parentheses to group conditions when you're combining both **and** and **or** operators to ensure the correct order of evaluation.

Short Circuit Evaluation of AND (and)

If the first condition is **False**, the operator stops testing. This is called short circuit evaluation. This can happen when the outcome of the expression is already determined based on the value of one of the operands, without needing to evaluate the other operand(s).

```
x = 5
y = 10

result = (x > 0) and (y < 0)
```

In this case, **x > 0** is **True**. **y < 0** is not evaluated because the outcome is already determined by the first operand being **True**. The value of **result** will be **False**.

Short Circuit Evaluation of OR (or)

When using the **or** operator, if the first operand is **True**, the overall expression will be **True** regardless of the value of the second operand. As a result, Python stops the evaluation early in this scenario as well.

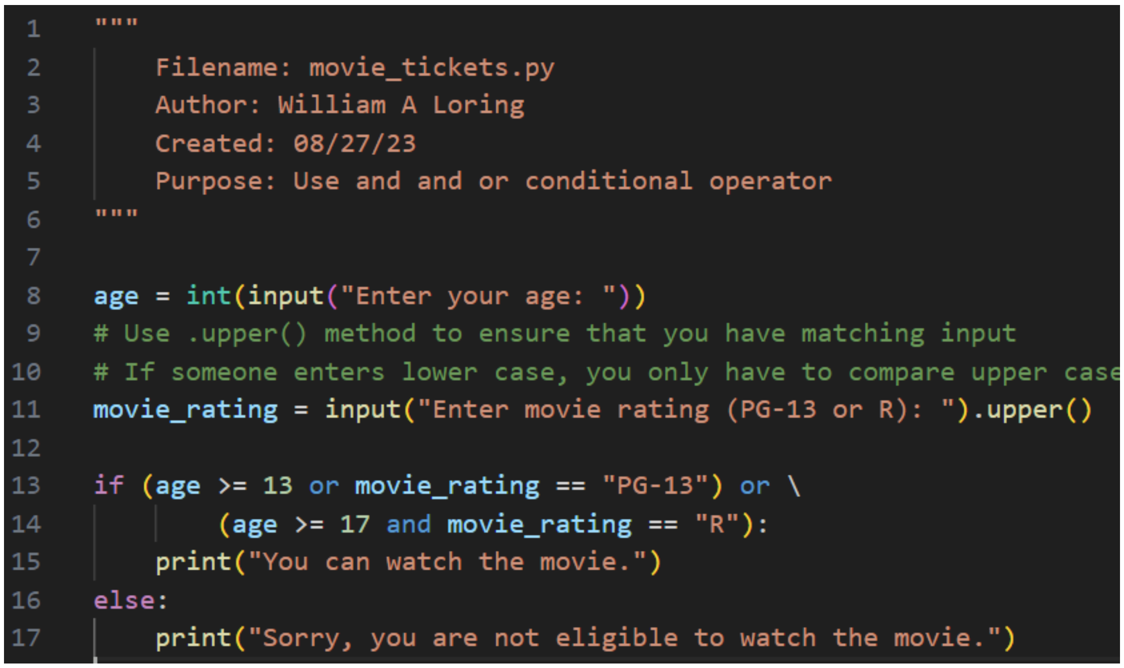
```
age = 15
has_permission = True

can_enter = (age >= 18) or has_permission
```

In this example, **age >= 18** is **False**, but **has_permission** is **True**. Since the first operand is **False** and the second operand is **True**, the overall expression is already determined to be **True** due to short-circuiting. The value of **can_enter** will be **True**.

Tutorial 3: Logical Operations And and Or

A movie theater has the following age restrictions for watching certain movies:

- To watch a PG-13 movie, a person must be at least 13 years old.
- To watch an R-rated movie, a person must be at least 17 years old.

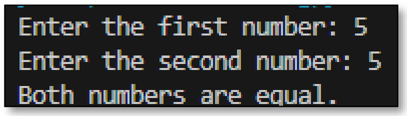


```python
"""
    Filename: movie_tickets.py
    Author: William A Loring
    Created: 08/27/23
    Purpose: Use and and or conditional operator
"""

age = int(input("Enter your age: "))
# Use .upper() method to ensure that you have matching input
# If someone enters lower case, you only have to compare upper case
movie_rating = input("Enter movie rating (PG-13 or R): ").upper()

if (age >= 13 or movie_rating == "PG-13") or \
        (age >= 17 and movie_rating == "R"):
    print("You can watch the movie.")
else:
    print("Sorry, you are not eligible to watch the movie.")
```

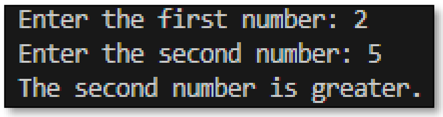
Assignment 1: Number Comparison

Write a Python program that takes two numbers as input and compares them. Print out the following messages based on the comparison:

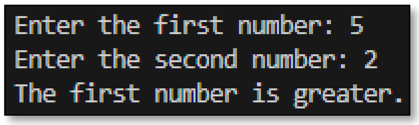
- If the first number is greater, print: "The first number is greater."
- If the second number is greater, print: "The second number is greater."
- If both numbers are equal, print: "Both numbers are equal."



```
Enter the first number: 5
Enter the second number: 5
Both numbers are equal.
```



```
Enter the first number: 2
Enter the second number: 5
The second number is greater.
```



```
Enter the first number: 5
Enter the second number: 2
The first number is greater.
```

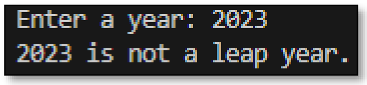
Assignment 2: Leap Year

Write a Python program that checks if a given year is a leap year or not.

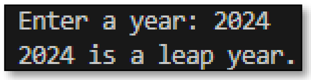
- If it is: divisible by 4: leap year
- Else if it is: divisible by 100: not leap year
- Else if it is: divisible by 400: leap year
- Otherwise it is: not a leap year

Create a Python program called **leap_year.py**

Example run:



```
Enter a year: 2023
2023 is not a leap year.
```



```
Enter a year: 2024
2024 is a leap year.
```

Assignment Submission

1. Use pseudocode or TODO.
2. Comment your code to show evidence of understanding.
3. Attach the program files.
4. Attach screenshots showing the successful operation of the program.
5. Submit in Blackboard.