

# ILP 2022 – W2S2

## If/Elif/Else statements

Matthieu DE MARI – Singapore University of Technology and Design



SINGAPORE UNIVERSITY OF  
TECHNOLOGY AND DESIGN

# Outline (Week2, Session2 – W2S2)

- The if statement
- The elif statement
- The else statement
- Dead code and code structure
- Nested ifs

# The **if** statement


The **if** statement is the simplest **conditional structure**.

# The **if** statement

The **if** statement is the simplest **conditional structure**.

- **Structure:**

- Use the keyword **if**,



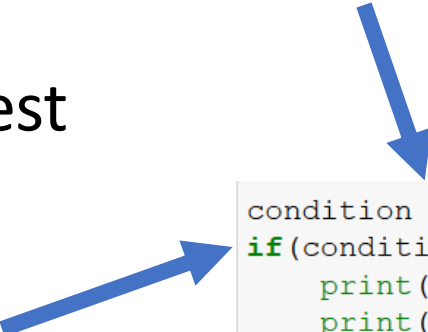
```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest conditional structure.

- **Structure:**

- Use the keyword **if**,
- Immediately after, pass a **Boolean variable** or write an **expression** that returns a **Boolean**,




```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest conditional structure.

- **Structure:**

- Use the keyword **if**,
- Immediately after, pass a **Boolean variable** or write an **expression that returns a Boolean**,
- Add a **colon symbol (:)** after the Boolean term,




```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest conditional structure.

- **Structure:**

- Use the keyword **if**,
- Immediately after, pass a **Boolean variable** or write an **expression that returns a Boolean**,
- Add a **colon symbol** (:) after the Boolean term,
- Add a block of instructions **inside** the **if** statement, which will be executed if and only if the Boolean is **True**.




```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest conditional structure.

- **Structure:**

- Use the keyword **if**,
- Immediately after pass a **Boolean variable** or write an **expression that returns a Boolean**,
- Add a **colon symbol** (:) after the Boolean term,
- Add a block of instructions **inside** the **if** statement, which will be executed if and only if the Boolean is **True**.



```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

**Note:** “inside” means your instruction is **indented** with 4 spaces more than the if statement. Jupyter will suggest indentations.



# The **if** statement

The **if** statement is the simplest **conditional structure**.

- **How it works:**
  - If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.

```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest **conditional structure**.



- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.


```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest **conditional structure**.

- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.




```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest conditional structure.

- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.



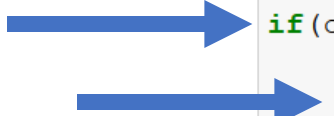
```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest **conditional structure**.

- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.



```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest **conditional structure**.

- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.




```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest conditional structure.

- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.



```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

# The **if** statement

The **if** statement is the simplest **conditional structure**.

- **How it works:**
  - If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.

```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

This will be printed if condition is set to True.  
It will not print if condition is set to False.  
This will be printed: not indented, outside of the if statement.



# The **if** statement

The **if** statement is the simplest conditional structure.

- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.
- If the Boolean condition is **False**, ignore the block of code in the **if** statement.

```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

This will be printed if condition is set to True.  
It will not print if condition is set to False.  
This will be printed: not indented, outside of the if statement.

```
condition = False # or True value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

This will be printed: not indented, outside of the if statement.

# The **if** statement

The **if** statement is the simplest conditional structure.

- **How it works:**

- If the Boolean condition/variable specified for the **if** statement is **True**, then execute the block of code inside the **if** statement.
- If the Boolean condition is **False**, ignore the block of code in the **if** statement.
- Once we are done executing the code in **if** (or ignoring it), move on to the next (non-indented) line.

```
condition = True # or False value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

This will be printed if condition is set to True.  
It will not print if condition is set to False.  
This will be printed: not indented, outside of the if statement.

```
condition = False # or True value
if(condition):
    print("This will be printed if condition is set to True.")
    print("It will not print if condition is set to False.")
print("This will be printed: not indented, outside of the if statement.")
```

This will be printed: not indented, outside of the if statement.

# The **elif** statement

The **elif** statement (short for “else-if”) is used to define another conditional test to be executed, if and only if the previous **if** statement has failed.

# The `elif` statement

The `elif` statement (short for “else-if”) is used to define another conditional test to be executed, if and only if the previous `if` statement has failed.

- **Structure:**

- Write your `if` block as before
- On the same indentation level as your `if` statement, write you `elif` statement (`elif` + Boolean condition + colon symbol)
- Add your instructions inside the `elif`, by indenting your code as in `if`.

```
bool1 = True # or False value
bool2 = True # or False value
if(bool1):
    print("Do something.")
elif(bool2):
    print("Do something else.")
```

# The **elif** statement

## How it works:

- If the Boolean in the **if** statement is **True**, execute the code inside the **if**, ignore the **elif**.

```
# Some booleans
bool1 = True
bool2 = True
# If statement, with True boolean condition
if(bool1):
    print("1. This will be printed, because bool1 is True.")
# Elif statement, with True boolean condition
elif(bool2):
    print("2. This will NOT be printed, because the first if block was executed.")
```

1. This will be printed, because bool1 is True.

# The **elif** statement

## How it works:

- If the Boolean in the **if** statement is **True**, execute the code inside the **if**, ignore the **elif**.

Did not  
execute  
despite  
bool2  
being  
True.

```
# Some booleans
bool1 = True
bool2 = True
# If statement, with True boolean condition
if(bool1):
    print("1. This will be printed, because bool1 is True.")
# Elif statement, with True boolean condition
elif(bool2):
    print("2. This will NOT be printed, because the first if block was executed.")

1. This will be printed, because bool1 is True.
```

# The **elif** statement

## How it works:

- If the Boolean in the **if** statement is **True**, execute the code inside the **if**, ignore the **elif**.
- Otherwise, check for the Boolean in **elif**, and execute the code indented inside the **elif**, if this second Boolean condition is **True**. Otherwise, ignore it.

```
# Some booleans
bool1 = False
bool2 = True
# If statement, with False boolean condition
if(bool1):
    print("1. This will NOT be printed, because bool1 is False.")
# Elif statement, with True boolean condition
elif(bool2):
    print("2. This will be printed, because the first if block was not executed and bool2 is True.")
```

2. This will be printed, because the first if block was not executed and bool2 is True.

# The **elif** statement (multiple blocks)

Multiple **elif** statements can be added after an initial **if** statement.

- In this case, execute the code inside an **elif**, if and only if:
  - all the previous **if/elif** have failed,
  - and its Boolean condition is **True**.

```
# Some booleans
bool1 = False
bool2 = True
bool3 = True
# If statement, with False boolean condition
if(bool1):
    print("1. This will NOT be printed, because bool1 is False.")
# Elif statement, with True boolean condition
elif(bool2):
    print("2. This will be printed, because the first if block was not executed and bool2 is True.")
# Another elif statement, with True boolean condition
elif(bool3):
    print("3. This will NOT be printed, because the previous block was executed.")
```

2. This will be printed, because the first if block was not executed and bool2 is True.



# The **elif** statement (multiple blocks)

Multiple **elif** statements can be added after an initial **if** statement.

- In this case, execute the code inside an **elif**, if and only if:
  - all the previous **if/elif** have failed,
  - and its Boolean condition is **True**.

```
# Some booleans
bool1 = False
bool2 = False
bool3 = True
# If statement, with False boolean condition
if(bool1):
    print("1. This will NOT be printed, because bool1 is False.")
# Elif statement, with False boolean condition
elif(bool2):
    print("2. This will NOT be printed, because bool2 is also False.")
# Another elif statement, with True boolean condition
elif(bool3):
    print("3. This will be printed, because none of the previous blocks were executed and bool3 is True.")
```

3. This will be printed, because none of the previous blocks were executed and bool3 is True.

# An example of **if/elif** code

**Example:** write a code that receives a number  $x$ , and prints one of the following prompts, accordingly:

- “ $x$  is strictly positive.”
- “ $x$  is strictly negative.”
- “ $x$  is zero.”

Let us use the **if/elif** structure to program that!

# An example of **if/elif** code

**Example:** write a code that receives a number  $x$ , and prints one of the following prompts, accordingly:

- “ $x$  is strictly positive.”
- “ $x$  is strictly negative.”
- “ $x$  is zero.”

Let us use the **if/elif** structure to program that!

```
# A number x
x = 10
# An if/elif/else statement
if(x>0):
    print("The number x is strictly positive.")
elif(x<0):
    print("The number x is strictly negative.")
elif(x==0):
    print("The number x is zero.")
```

The number  $x$  is strictly positive.

# Dead code

- **Definition (dead code):**  
We call “**dead code**” a piece of code that was written, but is never going to be executed.  
Often, due to bad structure in code.

# Dead code

- **Definition (dead code):**  
We call “**dead code**” a piece of code that was written, but is never going to be executed.  
Often, due to bad structure in code.
- **Question:** can you spot the line, which will never be executed, no matter what the value of **x** is?
- Why is it dead code?

```
if (x > 10) :  
    print("Hello!")  
elif (x > 12) :  
    print("World!")
```

# Dead code

- **Definition (dead code):**  
We call “**dead code**” a piece of code that was written, but is never going to be executed.  
Often, due to bad structure in code.
- **Question:** can you spot the line, which will never be executed, no matter what the value of **x** is?
- Why is it dead code?

```
if (x > 10) :  
    print("Hello!")  
elif (x > 12) :  
    { print("World!")
```

Dead  
code

# Dead code

- **Definition (dead code):**  
We call “**dead code**” a piece of code that was written, but is never going to be executed. Often, due to bad structure in code.
- **Question:** can you spot the line, which will never be executed, no matter what the value of **x** is?
- Why is it dead code?

```
if (x > 10) :  
    print("Hello!")  
elif (x > 12) :  
    { print("World!")
```

Dead code

**Reason:** Variable **x** cannot be both lower than 10 and greater than 12.

We need the **if** block to fail, for the **elif** to be checked. It means **x** must be lower than 10. But then, passing the Boolean condition in the **elif** requires having **x** greater than 12, but **x** will already be lower than 10 at this point.

# Dead code

- **Definition (dead code):**  
We call “**dead code**” a piece of code that was written, but is never going to be executed.  
Often, due to bad structure in code.

**Dead code** is often due to **bad structure/design** in the code.

Be careful!



```
if (x > 10) :  
    print("Hello!")  
elif (x > 12) :  
    print("World!")
```



# The **else** statement

The **else** statement is used to define a block of code to execute, if and only if **ALL** the previous **if/elif** statements have failed.

# The **else** statement

The **else** statement is used to define a block of code to execute, if and only if **ALL** the previous **if/elif** statements have failed.

Same structure as an **elif**, but...

- **Comes last**, after all the **if/elif** statements.
- **No Boolean condition** to be checked.

# The **else** statement (no **elif** example)

The **else** statement is used to define a block of code to execute, if and only if **ALL** the previous **if/elif** statement have failed.

Same structure as an **elif**, but...

- **Comes last**, after all the **if/elif** statements.
- **No Boolean condition** to be checked.

```
1 bool1 = True
2 if(bool1):
3     print("1. This will be printed, because bool1 is True.")
4 else:
5     print("2. This will NOT be printed, because the previous block was executed.")
```

1. This will be printed, because bool1 is True.

```
1 bool1 = False
2 if(bool1):
3     print("1. This will NOT be printed, because bool1 is False.")
4 else:
5     print("2. This will be printed, because none of the previous blocks were executed.")
```

2. This will be printed, because none of the previous blocks were executed.

# The **else** statement (multiple **elif** example)

```
1 bool1 = True
2 bool2 = True
3 if(bool1):
4     print("1. This will be printed, because bool1 is True.")
5 elif(bool2):
6     print("2. This will NOT be printed, because the previous block was executed.")
7 else:
8     print("3. This will NOT be printed, because the first block was executed.")
```

1. This will be printed, because bool1 is True.

```
1 bool1 = False
2 bool2 = True
3 if(bool1):
4     print("1. This will NOT be printed, because bool1 is False.")
5 elif(bool2):
6     print("2. This will be printed, because the first block was not executed and bool2 is True.")
7 else:
8     print("3. This will NOT be printed, because the second block was executed.")
```

2. This will be printed, because the first block was not executed and bool2 is True.

```
1 bool1 = False
2 bool2 = False
3 if(bool1):
4     print("1. This will NOT be printed, because bool1 is False.")
5 elif(bool2):
6     print("2. This will NOT be printed, because bool2 is False.")
7 else:
8     print("3. This will be printed, because none of the previous blocks were executed.")
```

3. This will be printed, because none of the previous blocks were executed.

Our previous **if/elif** example, turned into an **if/elif/else** example

```
1 # A number x
2 x = 10
3 # An if/elif/else statement
4 if(x>0):
5     print("The number x is strictly positive.")
6 elif(x<0):
7     print("The number x is strictly negative.")
8 elif(x==0):
9     print("The number x is zero.")
```

The number x is strictly positive.

# Our previous **if/elif** example, turned into an **if/elif/else** example

```
1 # A number x
2 x = 10
3 # An if/elif/else statement
4 if(x>0):
5     print("The number x is strictly positive.")
6 elif(x<0):
7     print("The number x is strictly negative.")
8 elif(x==0):
9     print("The number x is zero.")
```

The number x is strictly positive.

```
1 # A number x
2 x = 10
3 # An if/elif/else statement
4 if(x>0):
5     print("The number x is strictly positive.")
6 elif(x<0):
7     print("The number x is strictly negative.")
8 else:
9     print("The number x is zero.")
```

The number x is strictly positive.

# Our previous `if/elif` example, turned into an `if/elif/else` example

```
1 # A number x
2 x = 10
3 # An if/elif/else statement
4 if(x>0):
5     print("The number x is strictly positive.")
6 elif(x<0):
7     print("The number x is strictly negative.")
8 elif(x==0):
9     print("The number x is zero.")
```

The number x is strictly positive.

```
1 # A number x
2 x = 10
3 # An if/elif/else statement
4 if(x>0):
5     print("The number x is strictly positive.")
6 elif(x<0):
7     print("The number x is strictly negative.")
8 else:
9     print("The number x is zero.")
```

The number x is strictly positive.

```
1 # A number x
2 x = -5
3 # An if/elif/else statement
4 if(x>0):
5     print("The number x is strictly positive.")
6 elif(x<0):
7     print("The number x is strictly negative.")
8 else:
9     print("The number x is zero.")
```

The number x is strictly negative.

```
1 # A number x
2 x = 0
3 # An if/elif/else statement
4 if(x>0):
5     print("The number x is strictly positive.")
6 elif(x<0):
7     print("The number x is strictly negative.")
8 else:
9     print("The number x is zero.")
```

The number x is zero.

# Practice activities for `if/elif/else`

Let us practice the `if/elif/else` concepts a bit, with two activities.

**Activity 1 – Ask for user's age.ipynb**

**Activity 2 - Strength to lifepoints.ipynb**



# Activity 1 – Ask for user's age

Write a function **ask\_user\_age()**, as described below.

- It **receives no parameters** and **returns no parameters**.
- It first **asks for the user to input its age**, and retrieves the info from the user.
- **If the age is negative** (0 included), the function should **print** a message that reads "Your age cannot be negative, it must be at least 1."
- **If the age given by the user is larger than 122** (oldest person on record, Jeanne Calment), then the **print** should display "I really doubt you are \_\_\_\_ years old..." with the blank filled accordingly.
- **Otherwise**, the function should print "Oh, you are \_\_\_\_ years old? That's cool!", with the **blank filled** accordingly.

# Activity 2 - Strength to lifepoints

Write a function **strength\_to\_lifepoints()**, according to the following requirements.

- This function **receives a single parameter, strength\_points**, which corresponds to the number of strength points our main character has, and - for simplicity - will only take integer values.
- This function **returns a single output, lifepoints**, which corresponds to the number of lifepoints our main character will have, based on its strength points.
- Our main character has a **base number of 50 lifepoints** (that means it has 50 lifepoints, by default, if its strength is zero).
- **For each strength point**, our hero will **gain 10 extra lifepoints**.
- If the main character has **at least 50 strength points**, it gains a **one-time bonus of 100 lifepoints**, on top of the lifepoints it already has.
- Finally, if the main character has **at least 100 strength points**, it gains another **one-time bonus of 50% extra lifepoints**, on top of all the lifepoints it already has and the previous bonuses.

# Nested **if** structures

## **Definition (nested **if** structure):**

A **nested **if** structure** is a structure which includes one or multiple **if** statement(s), inside another **if** statement.

# Nested **if** structures

## Definition (nested **if** structure):

A **nested if structure** is a structure which includes one or multiple **if** statement(s), inside another **if** statement.

These are typically used to check additional conditions, based on whether another condition has been satisfied or not.

```
1 x = 5
2 if(x>=0):
3     print("The number x is positive.")
4     if(x>0):
5         print("In fact, the number x is STRICTLY positive.")
```

The number x is positive.  
In fact, the number x is STRICTLY positive.

```
1 x = 0
2 if(x>=0):
3     print("The number x is positive.")
4     if(x>0):
5         print("In fact, the number x is STRICTLY positive.")
```

The number x is positive.

# Nested **if** structures

## Definition (nested **if** structure):

A **nested if structure** is a structure which includes one or multiple **if** statement(s), inside another **if** statement.

These are typically used to check additional conditions, based on whether another condition has been satisfied or not.

Each **if** might have its own **elif/else** statements, placed on the same indentation level.

```
1 x = 5
2 if(x>=0):
3     print("The number x is positive.")
4     if(x>0):
5         print("In fact, the number x is STRICTLY positive.")
```

The number x is positive.  
In fact, the number x is STRICTLY positive.

```
1 x = 0
2 if(x>=0):
3     print("The number x is positive.")
4     if(x>0):
5         print("In fact, the number x is STRICTLY positive.")
```

The number x is positive.

```
1 x = -2
2 if(x>=0):
3     print("The number x is positive.")
4     if(x>0):
5         print("In fact, the number x is STRICTLY positive.")
6 else:
7     print("The number x is NOT positive.")
```

The number x is NOT positive.

# Nested **if** structures

```
1  x = 5
2  if (x==0):
3      print("The number x is zero.")
4  elif (x>=0):
5      print("The number x is positive.")
6      if (x>0):
7          print("In fact, the number x is STRICTLY positive.")
8  else:
9      print(("The number x is negative. "))
10     if (x<0):
11         print("In fact, the number x is STRICTLY negative.")
```

The number x is positive.

In fact, the number x is STRICTLY positive.

# Activity 3 - Race and class check

Write a function **character\_creation()**, according to the following requirements.

- The function will **receive two parameters**: **user\_race** and **user\_class**.
- For simplicity, only **three races** are available: **Human**, **Elf**, and **Dwarf**.
- For simplicity, **only four classes** are available: **Warrior**, **Hunter**, **Mage** and **Priest**.
- **Humans** can play **all classes**.
- **Elves** cannot be **warriors**.
- **Dwarves** cannot be **magicians or priests**.
- The function should **not return anything**.
- It should **print** "You cannot play a character that is ...{race} and ...{class}.", with **blanks filled** accordingly, **if the combination of user\_race and user\_class is not acceptable**.
- Not acceptable here means that its race and/or class is not among the ones listed above, or the combination is not permitted, as listed above.
- **If the combination is valid**, it should **print** "Your character's race is ...{race} and your character's class is ...{class}.", with blanks filled accordingly.

# Conclusion

- The if statement
- The elif statement
- The else statement
- Dead code and code structure
- Nested ifs



Up for a challenge?  
(in the Extra challenges folder)

**Activity 2+ - Strength to lifepoints (extra challenge).ipynb**

- Redo the activity 2, but this time...
- Do not use any conditional statement (**if/while**)
- The function should only contain one line, which starts with **return**.

# Nested **if** structures vs. combined conditionals

Nested **if** structures can, most of the time, be rewritten with combined conditionals (using **and/or** Boolean operators).

```
1 x = 5
2 if(x==0):
3     print("The number x is zero.")
4 elif(x>=0):
5     print("The number x is positive.")
6     if(x>0):
7         print("In fact, the number x is STRICTLY positive.")
8 else:
9     print(("The number x is negative. "))
10    if(x<0):
11        print("In fact, the number x is STRICTLY negative.")
```

The number x is positive.

In fact, the number x is STRICTLY positive.

# Nested **if** structures vs. combined conditionals

Nested **if** structures can, most of the time, be rewritten with combined conditionals (using **and/or** Boolean operators).

For instance, both structures on the right are equivalent.

```
1 x = 5
2 if(x==0):
3     print("The number x is zero.")
4 elif(x>=0):
5     print("The number x is positive.")
6     if(x>0):
7         print("In fact, the number x is STRICTLY positive.")
8 else:
9     print(("The number x is negative. "))
10    if(x<0):
11        print("In fact, the number x is STRICTLY negative.")
```

The number x is positive.

In fact, the number x is STRICTLY positive.

```
1 x = 5
2 if(x==0):
3     print("The number x is zero.")
4 if(x != 0 and x>=0):
5     print("The number x is non-zero and positive.")
6 if(x>0):
7     print("In fact, the number x is STRICTLY positive.")
8 if(x != 0 and x<=0):
9     print(("The number x is non-zero and negative. "))
10 if(x<0):
11     print("In fact, the number x is STRICTLY negative.")
```

The number x is non-zero and positive.

In fact, the number x is STRICTLY positive.

# Nested **if** structures vs. combined conditionals

Nested **if** structures can, most of the time, be rewritten with combined conditionals (using **and/or** Boolean operators).

For instance, both structures on the right are equivalent.

**Personal preference:** Whenever possible, try to avoid the nested **if** structures. They are often overly complicated and prone to errors in designing the code.

```
1 x = 5
2 if(x==0):
3     print("The number x is zero.")
4 elif(x>=0):
5     print("The number x is positive.")
6     if(x>0):
7         print("In fact, the number x is STRICTLY positive.")
8 else:
9     print(("The number x is negative. "))
10    if(x<0):
11        print("In fact, the number x is STRICTLY negative.")
```

The number x is positive.

In fact, the number x is STRICTLY positive.

```
1 x = 5
2 if(x==0):
3     print("The number x is zero.")
4 if(x != 0 and x>=0):
5     print("The number x is non-zero and positive.")
6 if(x>0):
7     print("In fact, the number x is STRICTLY positive.")
8 if(x != 0 and x<=0):
9     print(("The number x is non-zero and negative. "))
10 if(x<0):
11     print("In fact, the number x is STRICTLY negative.")
```

The number x is non-zero and positive.

In fact, the number x is STRICTLY positive.