Python for Data Analysis: Methods & Tools

Python for everyday people

Written by Felipe Dominguez - Professor Adjunct
Hult International Business School

Chapter 03 - Conditional Statements

If true else false

1. The logic behind conditions

Congratulations on learning the basics of Python! You now know how to print, ask for user input, seek help, and use different data types. This puts you in a good position to start creating more complex codes and programs. In this chapter, you will be introduced the concept of **conditional statements** and how you can use them in your programs.

Conditional statements are conditions that determine a certain outcome based on **whether they are true or false**. There are many real-world situations that can be represented as conditional statements. For example:

```
If there is a snowstorm the flight will be canceled.
Otherwise, the flight will operate as scheduled.
```

As you can see, the statement above begins with the word **if** followed by a condition (there is a snowstorm). If the condition is true, a certain action will be taken (the flight will be canceled). If the condition is false (Otherwise) a different action is triggered (the flight will operate as scheduled).
As in the real world, Python conditional statements start with **if** whereas **else** is used to state the the alternative statement (Otherwise). Conditional statements, as the real-world, can quickly become more complex, as in the following example:

```
If there is a snowstorm the flight will be canceled.
The airline will send an email to all passengers with the new
flight time.
```

```
We will also issue a refund and compensation for those
passengers who are unable to catch the new flight.
For those with connections, we will re-schedule their
connections to match the new flight time.
Finally, we will identify our airline members and issue a
voucher for a stay at the nearest hotel.
Otherwise, the flight will operate as scheduled
```

2. Understanding Conditionals

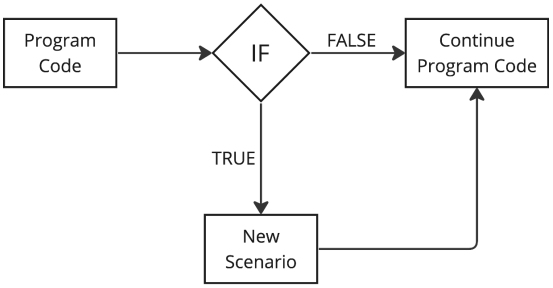

Figure 2.1. Conceptual drawing of Conditional Statement

Figure 2.1 illustrates a conceptual drawing of **conditional statements**. As shown, **if** a condition is **true**, Python will execute a specific set of code (referred to as a new scenario), whereas if the condition is **False** it will move on to the rest of the program. The **elif** and **else** provide additional options for triggering different scenarios based on the truth of the conditions before continuing with the program code.

Whenever Python encounters a conditional statement, it makes a decision based on the conditions provided. It is important to understand that Python is simply following a set of rules and making decisions based on these conditions.

Note: *While conditional statements were initially used in the development of artificial intelligence (along with statistics & probabilities), this does not mean that they are AI themselves. When you provide a set of conditional statements to Python, you are simply giving it pre-written instructions, rather than allowing it to learn and think independently.*

2.1. Triggering different scenarios

Let's begin by examining the simplest form of a conditional statement and understand the meaning of each part of the code. Code 2.1.1 shows an **if** statement. Let's take a closer look at the code.

#	Code

1	|	if cars > drivers:
2	|	print("We don't have enough drivers!")
3	|	print("This print is out of the if")

Table 2.1.1. If statement.

As you can see on line 1, the first word of the statement is **if** and ends with a **semicolon (:)**. Python understand that anything between the if and the semicolon is the condition you want to check (either true or false).

Indentation matters! Notice the indentation before the print statement on code 2.1.1. This tells Python that this piece of code is within the if statement. Therefore, Python will execute this code only if the condition (cars > drivers) is True. Anything outside of the indentation, like the last line of the code, is outside of the conditional statement. Hence, Python will run it regardless of whether the condition is True or False.

In [1]:
```
# Code 2.1.1.

# Declare variables
drivers = 20

cars  = 100

# Creating conditional statement
if cars > drivers:
    print("We don't have enough drivers!")

print("This print is out of the if")
```

```
We don't have enough drivers!
This print is out of the if
```

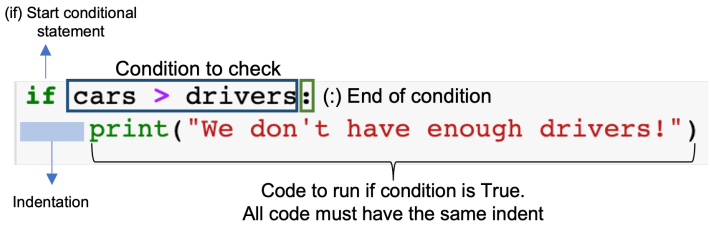

Figure 2.1.1. Conditional Statement

2.2. If and Else Python syntax

Code 2.1.1. shows the basic conditional statement, where Python evaluates one condition and run a code if that condition is True. But what if you want to run a different code if the condition is False?

For example, if you have more drivers than cars? In this case, you can use the **else** clause. The 'else' clause runs when the condition given in the 'if' statement is not met (i.e. is False).

Scenario 2 in Figure 2.1 represents this situation. Code 2.2.2 shows a conditional statement with **if** and **else** statements.

In [2]:
```
# Code 2.2.1.

# Declare variables
drivers = 150
cars  = 100

# Creating conditional statement
if cars > drivers:
    print("We don't have enough drivers!")
else:
    print("We don't have enough cars!")
```

```
We don't have enough cars!
```

2.3. More conditions, more actions!

Now that you know how to evaluate one condition and run different codes depending on whether it is True or False, let's consider the case where you want to evaluate multiple conditions or execute different outcomes depending on specific situations. Well, this can be achieved using the **elif** (else if) statement. 'Elif' allows you to add additional scenarios to your code if the initial condition is not met. Figure 2.3 illustrates an example with 3 scenarios:

- Scenario 1: Initial condition is True
- Scenario 2: Initial condition is False, but elif condition is True
- Scenario 3: Neither the initial nor elif conditions are True

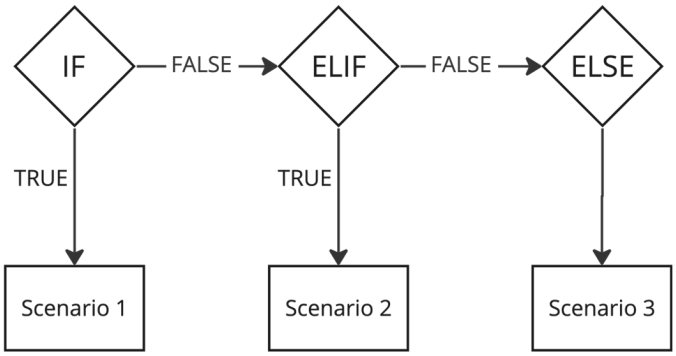

Figure 2.3. Conceptual drawing of a Conditional Statement

Elif and **else** statements are at the same indentation level as **if**. This is how Python understands the order of conditions. First, it checks the initial condition. If the condition is False, it checks any 'elif' statements that follow. If all 'elif' statements are False, it moves to the 'else' statement. It is important to understand this logic, as Python follows it strictly. Figure 2.4. present a conceptual drawing conditional statement structure with n conditions.

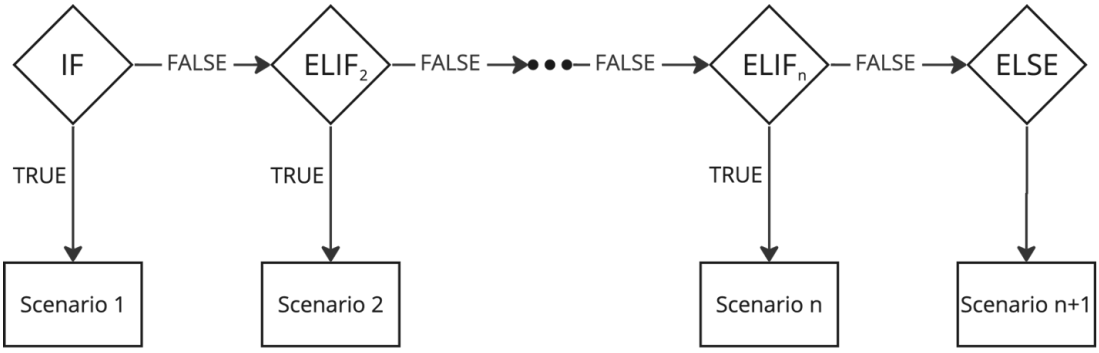

Figure 2.4. Conceptual drawing of a full Conditional Statement

Additionally, Python will run the code for the first condition that is met. Therefore, if both the 'if' and 'elif' conditions are True, Python will only run the code for the 'if' statement. Let's examine code 2.3.1 and 2.3.2.

In [1]:
```
# Code 2.3.1.

# Declare variables
drivers = 100
cars  = 100

# Creating conditional statement
if cars > drivers:                              # Evaluate this condition first
    print("We don't have enough drivers!")

elif cars == drivers:                           # Evaluate this condition secon
    print("There are enough drivers and cars!")

else:                                           # If all conditions are false,
    print("We don't have enough cars!")
```

```
There are enough drivers and cars!
```

In [4]:
```
# Code 2.3.2.
# Check how the output of this code is different than 2.3.1.
# This is because Python checks the condition on the if statement (line 11)
# before the elif statement (line 13)

# Declare variables
drivers = 100
cars  = 100
```

```
# Creating conditional statement
if cars >= drivers:                              # Evaluate this condition firs
    print("We don't have enough drivers!")

elif cars == drivers:                           # Evaluate this condition secon
    print("There are enough drivers and cars!")

else:                                           # If all conditions are false,
    print("We don't have enough cars!")
```

```
We don't have enough drivers!
```

2.4 Else as an except controller

The **else** statement is a useful tool in Python, as it allows you to handle unexpected situations and provide some control over your program. For example, if you ask the user to select a value from a list or set, you can double-check that the user made the correct selection and send a message if they did not. Take a look at codes 2.4.1. and 2.4.2.

Note: *here are other ways to control and identify unexpected behaviors in your code, such as* ***try/exempt****. You will review this in class 6.*

In [2]:
```
# Code 2.4.1.

# Declare set of car brands
my_cars = {"Porsche", "Ferrari", "BMW"}

# Declare user input
user_car = input(prompt = f"""
What is your favorite car brand?
Select one from {my_cars} - """)

# Check if user select a car within the set.
if user_car in my_cars:
    print(f"""
Wow! {user_car} is a fantastic brand!""")

else:
    print("""
I couldn't find the brand you mentioned \
in my database.
Please try again""")
```

```
What is your favorite car brand?
Select one from {'BMW', 'Porsche', 'Ferrari'} - Mazda

I couldn't find the brand you mentioned in my database.
Please try again
```

In [19]:
```
# Code 2.4.2.
```

```
# Ask user input
user_select = input(prompt = f"""
Which NFL team has the most Super Bowl titles?
Please select 1, 2, or 3)
1. New England Patriots
2. Pittsburgh Steelers
3. San Francisco 49ers""")

# Declare multiple conditional statement
if user_select == '1':
    print("""\nPerfect! New England Patriots and
Pittsburgh Steelers have 6 Super Bowl title each.""")

elif user_select == '2':
    print("""\nPerfect! New England Patriots and
Pittsburgh Steelers have 6 Super Bowl title each.""")

elif user_select == '3':
    print("That's not correct! Try again!")

else:
    print("Something went wrong, please try again.")
```

```
Which NFL team has the most Super Bowl titles?
Please select 1, 2, or 3)
1. New England Patriots
2. Pittsburgh Steelers
3. San Francisco 49ers2

Perfect! New England Patriots and
Pittsburgh Steelers have 6 Super Bowl title each
```

Comments on code 2.4.1. and 2.4.2.

Codes 2.4.1. and 2.4.2. attempt to control unexpected outputs or bugs. The input prompt clearly specify what the user should input (either the number of cars or a value from the list 1, 2, or 3). Additionally, the 'if' and 'elif' statements in code 2.4.2 take into account that Python will store the user's input as a string, so they check each condition accordingly. Finally, both codes have an 'else' clause to catch any invalid user input or unexpected bugs and communicate it to the user. However, this may not be sufficient. For example, what if the user inputs the name of the team in code 2.4.2, or provides answers such as 'NEW ENGLAND PATRIOTS' or 'New EnGlAND Patriots'? These answers should trigger the correct output, but the program currently cannot understand them.

Well, there are ways to make your conditional statements more flexible and provide a better user experience, while also reducing the possibility of bugs or errors in your code.

3. Booleans and conditional statements

In this section you will practice conditional statements based on a combination of booleans conditions. Remember that conditional statements run a code based on the value of the condition, which always return a True or False value (Boolean type)

3.1. Numerical conditions

As you can imagine by now, you can set numerical conditions for conditional statements. Therefore, you can leverage mathematical operations to return True or False regarding numerical values.

In [25]:
```
# Code 3.1.1.

# import random package
import random

# set random seed
random.seed(1)

# declare python number
python_number = random.randint(10,100)

# Declare user input
user_number = input(prompt = "Can you guess my number? \nSelect a number betwe
user_number = int(user_number)

if user_number == python_number:
    print(f"""\nYou genius! That's correct, my number is {python_number}""")

elif user_number > python_number:
    difference = abs(python_number - user_number)
    print(f"""\nMmm... You were high by {difference}""")

elif user_number < python_number:
    difference = abs(python_number - user_number)
    print(f"""\nMmm... You were low by {difference}""")

else:
    print("\nPlease select a number between 10 and 100")
```

```
Can you guess my number?
Select a number between 10 and 100 20

Mmm... You were low by 7
```

In [3]:
```
# Code 3.1.2.

# Let's practice booleans conditions with numbers

# Declare list of car brands
my_cars = ["Porsche", "Ferrari", "Mazda", "BMW"]

# Conditional statement
if len(my_cars) >= 10:
    print("Wow! You have a vast collection of cars!")
elif len(my_cars) < 10 and len(my_cars) >1:
```

```
    print("You are bulding your collection of cars!")
else:
    print("It is always a good moment to start your car collection!")
```

```
You are bulding your collection of cars!
```

3.2. String conditions

Imagine you ask the user to input a city and you want to provide a description of the city based on their selection. Even if you clearly describe the options to the user, they may input something slightly different. For example, the user may input "AMSTERDAM", "Amsterdam", "amsterdam", "city of Amsterdam", or other variations. All of these answers are correct, but Python will see them as different answers (see code 3.2.1). Keep in mind that each letter has a unique address in Python.

There are two steps you can take to make your code more flexible with user input:

```
i. Transform the user's input into the same string format as
the condition (lowercase, uppercase, capitalized, title)
ii. Create a condition to check if a key word is within user's
input
```

In [7]:
```
# Code 3.2.1.

# Declare variable city
city = "Amsterdam"

# Check and print conditions
print(f"""Is {city} == to "amsterdam"? {city == "amsterdam"}""")
print(f"""Is {city} == to "AmSteRdam"? {city == "amsterdam"}""")
```

```
Is Amsterdam == to "amsterdam"? False
Is Amsterdam == to "AmSteRdam"? False
```

3.2.1. Leverage string methods (lower, upper, and capital case)

As previously mentioned, the user may input text in any format, and you cannot assume that they will act as you expect. Therefore, you need to have some control over their inputs. Fortunately, Python has a variety of string methods that you can use to prevent misleading answers from the user.

Table 3.2.1 lists the most commonly used string methods.

Method	Description
.capitalize()	capitalizes the first character of a string, leaving the rest lowercase
.casefold()	lowercases all characters in a string (*includes* special characters)

.lower()	lowercases all characters in a string (*excludes* special characters)
.upper()	capitalizes all characters in a string
.title()	capitalizes first letter of each word

Table 3.2.1. String methods

In [6]:
```
# Code 3.2.1.

# Declare city variable
city = "Amsterdam is awesome (ß)!"

# Print string methods
print(f"""{"-" * 110}
- Capitalize City: {city.capitalize()}
- Case fold city: {city.casefold()}
- Lower case City: {city.lower()}
- Upper case City: {city.upper()}
- Title City: {city.title()}
{"-" * 110}""")
```

```
--------------------------------------------------------------------------------
---------------------------------
- Capitalize City: Amsterdam is awesome (ß)!
- Case fold city: amsterdam is awesome (ss)!
- Lower case City: amsterdam is awesome (ß)!
- Upper case City: AMSTERDAM IS AWESOME (SS)!
- Title City: Amsterdam Is Awesome (Ss)!
--------------------------------------------------------------------------------
---------------------------------
```

In [50]:
```
# Code 3.2.2.

# Ask user to input a city.
user_city = input(prompt = """Between this three cities, which one is your fa
1. Amsterdam
2. Santiago
3. Berlin
""")

# transform user's answer into casefold.
user_city = user_city.casefold()

# Conditional statement
if user_city == "amsterdam":
    print("""
Did you know...
Amsterdam is the capital and most populous city of the Netherlands, with
The Hague being the seat of government""")

elif user_city == "santiago":
    print("""
Did you know...
Santiago, also known as Santiago de Chile, is the capital and largest city of
Chile as well as one of the largest cities in the Americas.""")

elif user_city == "berlin":
    print("""
Did you know...
```

```
Berlin is the capital and largest city of Germany by both area and population
Its 3.6 million inhabitants make it the European Union's most populous city,
according to population within city limits""")

else:
    print("Something went wrong. Please try again")
```

```
Which city is your favorite?
1. Amsterdam
2. Santiago
3. Berlin
SANTIago

Santiago, also known as Santiago de Chile, is the capital and largest city of
Chile as well as one of the largest cities in the Americas.
```

3.3. "In" conditions

But what if the user provides a slightly different answer, such as:

```
Mmm... I'm not sure, but I like Santiago
```

In this case, the user did not select a specific city. However, the program should be flexible enough to understand that the user is indicating a preference for "Santiago". Python can help you by using the **in** statement, which checks if a specific word (key word) is present within a set and returns either True or False. This can be used as a new condition in your 'if' statement.

Let's look at code 3.2.3.

In [10]:
```
# Code 3.2.3.

# Ask user to input a city.
user_city = input(prompt = """Between this three cities, which one is your fa
1. Amsterdam
2. Santiago
3. Berlin
""")

# transform user's answer into casefold.
user_city = user_city.casefold()

# Conditional statement
if user_city == "1":
    print("""
Did you know...
Amsterdam is the capital and most populous city of the Netherlands, with
The Hague being the seat of government""")

elif "amsterdam" in user_city:
    print("""
Did you know...
Amsterdam is the capital and most populous city of the Netherlands, with
```

```
The Hague being the seat of government""")

elif user_city == "2":
    print("""
Did you know...
Santiago, also known as Santiago de Chile, is the capital and largest city of
Chile as well as one of the largest cities in the Americas.""")

elif "santiago" in user_city:
    print("""
Did you know...
Santiago, also known as Santiago de Chile, is the capital and largest city of
Chile as well as one of the largest cities in the Americas.""")

elif user_city == "3":
    print("""
Did you know...
Berlin is the capital and largest city of Germany by both area and population
Its 3.6 million inhabitants make it the European Union's most populous city,
according to population within city limits""")

elif "berlin" in user_city:
    print("""
Did you know...
Berlin is the capital and largest city of Germany by both area and population
Its 3.6 million inhabitants make it the European Union's most populous city,
according to population within city limits""")

else:
    print("Something went wrong. Please try again")
```

```
Between this three cities, which one is your favorite?
1. Amsterdam
2. Santiago
3. Berlin
I don't know.. I would say Berlin

Did you know...
Berlin is the capital and largest city of Germany by both area and populatio
n.
Its 3.6 million inhabitants make it the European Union's most populous city,
according to population within city limits
```

3.4. Boolean combinations (and/or)

Code 3.2.3 effectively controls the user's input, but it may seem redundant as it includes two conditions per city that return the same output. You can use boolean combination syntax **(and/or)** to consolidate your code. Take a look at code 3.2.3 and observe how you can combine multiple conditions into a single line.

In [11]:
```
# Code 3.2.3.

# Ask user to input a city.
user_city = input(prompt = """Between this three cities, which one is your fa
1. Amsterdam
```

```
2. Santiago
3. Berlin
""")

# transform user's answer into casefold.
user_city = user_city.casefold()

# Conditional statement
if user_city == "1" or "amsterdam" in user_city:
    print("""
Did you know...
Amsterdam is the capital and most populous city of the Netherlands, with
The Hague being the seat of government""")

elif user_city == "2" or "santiago" in user_city:
    print("""
Did you know...
Santiago, also known as Santiago de Chile, is the capital and largest city of
Chile as well as one of the largest cities in the Americas.""")

elif user_city == "3" or "berlin" in user_city:
    print("""
Did you know...
Berlin is the capital and largest city of Germany by both area and population
Its 3.6 million inhabitants make it the European Union's most populous city,
according to population within city limits""")

else:
    print("Something went wrong. Please try again")
```

```
Between this three cities, which one is your favorite?
1. Amsterdam
2. Santiago
3. Berlin
I don't know.. I would say Berlin

Did you know...
Berlin is the capital and largest city of Germany by both area and populatio
n.
Its 3.6 million inhabitants make it the European Union's most populous city,
according to population within city limits
```

4. Nested conditionals

You are almost done! Let's discuss about **nested conditionals**. This term refers to the use of conditionals statements within other conditional statements. Nested conditionals allows you to program more complex programs for more complex situations. You might want to check a first condition and run a code that has multiple conditions within it. Figure 2.5.

illustrates a nested conditional structure.

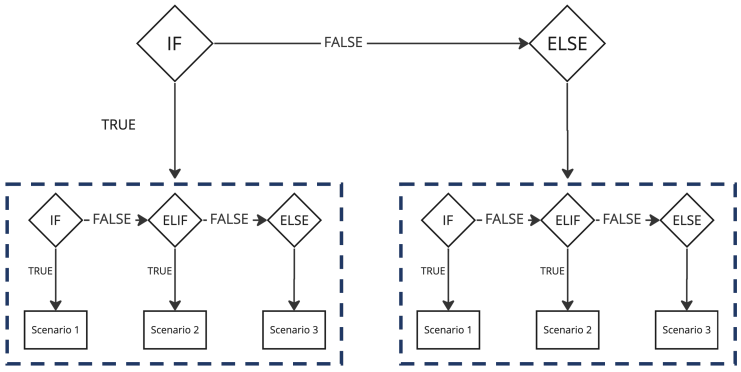

Figure 2.5. Conceptual structure of nested conditional statements

Let's take a look at code 4.1. and 4.2. which includes examples of nested conditionals.

In [58]:
```
# Code 4.1.

# import random package
import random

# set random seed
random.seed(1)

# declare python number
python_number = random.randint(10,100)

# Declare user input
user_number = input(prompt = "Can you guess my number? \nSelect a number betwe
user_number = int(user_number)

if user_number == python_number:
    print(f"""\nYou genius! That's correct, my number is {python_number}""")

elif user_number != python_number:

    # Calculate difference between python and user numbers
    difference = (python_number - user_number)

    if difference > 0: # Nested conditional
        print(f"""\nMmm... You were low by {abs(difference)}""")

    elif difference < 0:
        print(f"""\nMmm... You were high by {abs(difference)}""")

else:
    print("\nPlease select a number between 10 and 100")
```

```
Can you guess my number?
Select a number between 10 and 100 24

Mmm... You were low by 3
```

In [14]:
```
# Code 4.2.

# Ask user for they supermarket list
```

```
sup_list = input(prompt = """Enter the list you usually buy at the supermarke
Separate each item by a comma and space(', '):  """)

# Transform user input into a list
sup_list = sup_list.split(", ")

# check condition if user input less than 5 item or not
if len(sup_list) >= 5:

    if len(sup_list) > 10:
        print("That's a big list!")

    else:
        print("That's a medium list")

else:
    print("That's a very short list")
```

```
Enter the list you usually buy at the supermarket
Separate each item by a comma and space(', '):  beef, meal, corn, rice, spagh
etti, juice, butter
That's a medium list
```

5. Summary

Congratulations! You feel proud of yourself! You now know how to work with conditional statements, you are on the way to build even more complex programs! See you on Chapter 04!