Lesson 4: More branches

Year 8 – Intro to Python programming

National Centre for Computing Education

Raspberry Pi

Strange weather

```
print("Where do you live?")
location = input()
print("Weather in", location, "now?")
weather = input()
```

How would you **extend** this program to give advice to the user on how to dress **depending** on the weather?

Provide a rough outline of what your Python code would look like.

Give advice for when the `weather` is `cloudy`, `rainy`, or `snowy`.

In any other case, display a generic message.

Discuss in pairs and write an answer together.

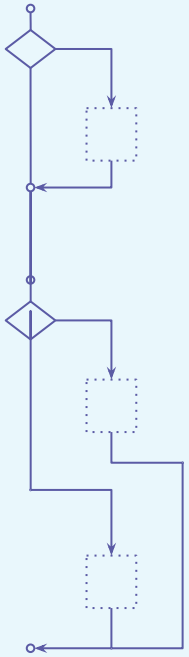
In this lesson, you will...

- Explore how **selection** can handle more than two possible **branches**
- Use **iteration** (`while` statements) to allow the flow of program execution to include **loops**

Selection

```
if condition :
    block of
    statements

if condition :
    block of
    statements
else:
    block of
    statements
```



These versions of selection check **one condition** and select one out of **two branches** to follow.

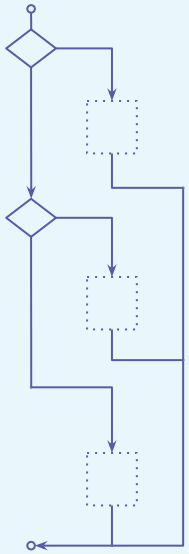
What if there are more than two cases in the problem?

What if there are multiple branches ?

Think of the weather example:
“Give advice for when the weather is cloudy, rainy, or snowy.”

Multi-branch selection

```
if condition :
    block of statements
elif condition :
    block of statements
else:
    block of statements
```



Multi-branch selection (`if`, `elif`, `else`) checks **successive conditions** and selects one out of **multiple branches** to follow.

You will need an `if` block with `elif` blocks:

when there are **more than two mutually exclusive paths** for your program to follow.

You can use multiple `elif` blocks.

The `else` and its block are optional.

You can **nest** anything inside the blocks of statements, even more `if` statements.

Strange weather

```
print("Where do you live?")
location = input()
print("Weather in", location, "now?")
weather = input()
```

Extend this program using `elif`, to give advice to the user on how to dress **depending** on the weather.

Give advice for when the `weather` is: `cloudy`, `rainy`, or `snowy`.

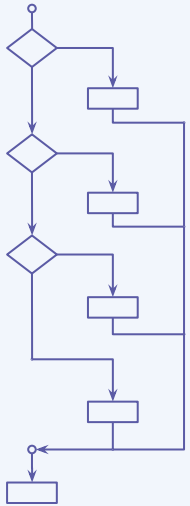
In any other case, display a generic message.

Live coding (ncce.io/py-weather-40)

Strange weather: sample solution

```
if weather == "cloudy":
  advice = "No sunglasses"
elif weather == "rainy":
  advice = "Get an umbrella"
elif weather == "snowy":
  advice = "Mittens and earmuffs"
else:
  advice = "No particular advice"

print(advice)
```



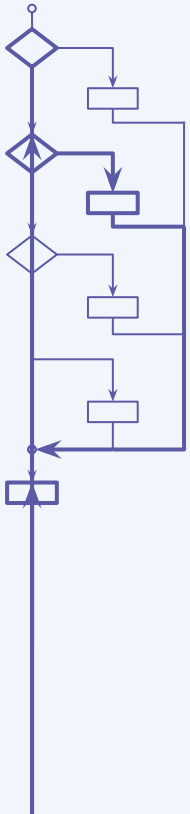
Strange weather: executing the program

```
if weather == "cloudy":
  advice = "No sunglasses"
elif weather == "rainy":
  advice = "Get an umbrella"
elif weather == "snowy":
  advice = "Mittens and earmuffs"
else:
  advice = "No particular advice"

print(advice)
```

False

True



State

weather	"rainy"
advice	"Get an umbrella"

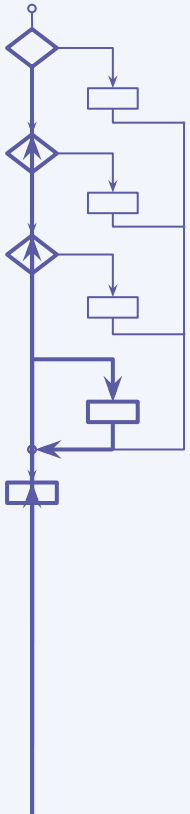
Output

```
Get an umbrella
```

Strange weather: executing the program

```
if weather == "cloudy":          False
  advice = "No sunglasses"
elif weather == "rainy":         False
  advice = "Get an umbrella"
elif weather == "snowy":         False
  advice = "Mittens and earmuffs"
else:
  advice = "No particular advice"

print(advice)
```



State

weather	"clear"
advice	"No particular advice"

Output

```
No particular advice
```

Retrieving the weather

```
from ncce.weather import description

print("Where do you live?")
location = input()
weather = description(location)
print("The weather is", weather)
```

This modified program retrieves the weather online, instead of asking the user.

Note The `weather` module and its functions are **not** standard Python components.

Open the modified program in your development environment (ncce.io/py-weather-42).

Run it a few times and enter different locations around the world.

People in space

You will be given a program that displays how many people are currently in space.

Note: The number is retrieved from an online service. It is not always the same.

Extend this program so that it asks the user to **guess** this number.

Use your **worksheet**.

Task People in space

Below is a short program that displays how many people are currently in space.

```
1 from ncce.space import people
2 number = people()
3 print(number, "people in space right now")
```

Line 1 imports the `people` function from the `space` module, in order to retrieve this information from an online service, so the number of people displayed will not always be the same. This is **not a standard Python component**; it has been created specifically to allow you to perform these tasks.

Step 1

Open this Python program (ncce.io/py-space-40) in your development environment and **extend** it, so that it asks the user to guess the number of people currently in space.

Example

Note: The number of people in space is retrieved from an online service through the `people` function. It is not always the same and the numbers shown here are just an example.

The program displays a prompt and waits for keyboard input.	`How many people do you think are in space right now?`
The user types in a reply.	`5`
The program displays the correct number.	`8 people in space right now`

People in space: sample solution

```
from space import people
number = people()
```

Retrieve the number of people in space

```
print("How many people...")
guess = int(input())
```

Prompt the user to guess

```
if guess < number:
  print("It's actually more than that.")
elif guess > number:
  print("It's actually less than that.")
else:
  print("That's right!")
```

Check the answer and provide feedback

```
print(number, "people in space now")
```

Display the number of people in space

Something missing

```
print("What's your name?")
name = input()
print("Hello", name)
```

This program greets the user by name.

What if...

we wanted to **repeat** the process until a specific name is entered?

Many of our programs have sections that could be repeated

e.g. checking the weather for multiple locations, and a number guessing game with many guesses.

Iteration

When your programs **repeat** actions, checking for a terminating **condition** at the beginning of each new loop

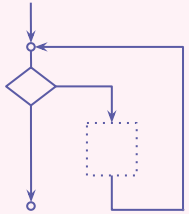
Iteration

```
while condition :
    block of
    statements
```


```
while True:
    block of
    statements
```

Iteration

```
while condition :
    block of
    statements
```



You will need a `while` statement:

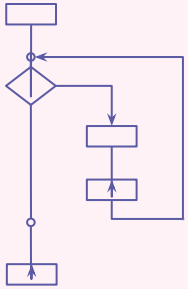
when your program needs to repeat actions while a condition is satisfied.

Example: a (surreal) iterative program

```
print("What's your name?")
name = input()

while name != "Hedy":
    print("Try again Hedy")
    name = input()

print("Hello", name)
```



Question Can you **predict** what the outcome of running this program will be?

Tip: `while` block These statements are to be repeated.

Tip: `while` condition This condition is checked at the beginning of each loop.

Answer This program will keep asking the user for their name, until the name is `"Hedy"`, at which point it will display a greeting (ncce.io/py-you-4).

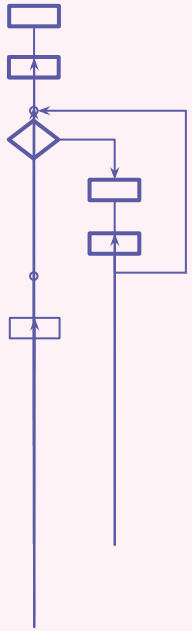
Iteration: step-by-step execution

```
print("What's your name?")
name = input()

while name != "Hedy":
    print("Try again Hedy")
    name = input()

print("Hello", name)
```

True



State

name "Margaret"

Input/Output

What's your name?

User types Margaret

Try again Hedy

User types Ada

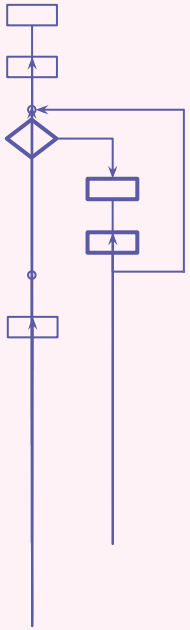
Iteration: step-by-step execution

```
print("What's your name?")
name = input()

while name != "Hedy":
    print("Try again Hedy")
    name = input()

print("Hello", name)
```

True



State

name "Ada"

Input/Output

What's your name?

User types Margaret

Try again Hedy

User types Ada

Try again Hedy

User types Hedy

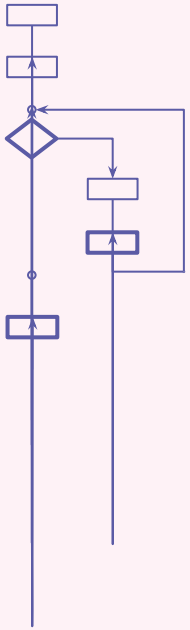
Iteration: step-by-step execution

```
print("What's your name?")
name = input()

while name != "Hedy":
  print("Try again Hedy")
  name = input()

print("Hello", name)
```

False



State

name "Hedy"

Input/Output

```
What's your name?
User types Margaret
Try again Hedy
User types Ada
Try again Hedy
User types Hedy
Hello Hedy
```

Subtle points

```
print("What's your name?")
name = input()

while name != "Hedy":
  print("Try again Hedy")
  name = input()

print("Hello", name)
```

Question

What difference will it make if the line in red is removed?

A 1. It will make no difference.
2. There's no initial value for `name`: an error will occur when checking the condition in `while`.
A 3. There's no initial value for `name`: an error will occur when executing `print`.
4. There's no initial value for `name`: the program will execute normally.

Subtle points

```
print("What's your name?")
name = input()

while name != "Hedy":
  print("Try again Hedy")
  name = input()

print("Hello", name)
```

Question

What difference will it make if the line in red is removed?

(Assume the first `name` is not `"Hedy"`.)

A
1. It will make no difference.
2. The value for `name` is never modified: an error will occur when checking the condition in `while`.
3. The value for `name` is not modified: the program will never terminate.

Answer the questions in your homework sheet.

In this lesson, you...

Explored how **selection** can handle more than two possible **branches**

Used **iteration** (`while` statements) to allow the flow of program execution to include **loops**

Next lesson, you will...

Use **iteration** (`while` statements) to allow the flow of program execution to include **loops**

Use **Boolean** variables, operators, and expressions

Use variables as **counters** and **flags**