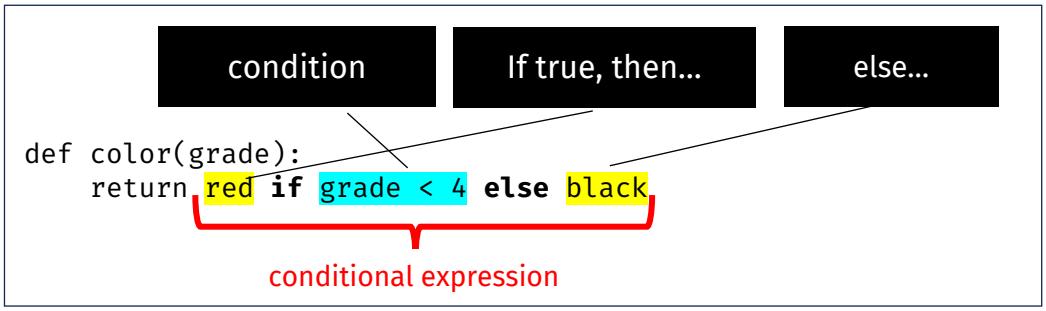
Name:	Date:

Selection

A program sometimes may have to **make choices**. For example, a program may have to show a grade in red if it is a failing grade, and in black if it is a passing grade.

In programming we call this operation **selection**.
We need to specify a **condition**—a logical expression that produces a truth value. If the condition is true, Python will take a certain path; otherwise, it will take another.

Here is an example:



In the above example, the condition uses the operator **<** ("less than") that we already know from mathematics.

How could we express the condition differently in an **equivalent** way?

Python provides several relational operators one can use to compare two numbers. Complete this table:

Operator	Meaning	Example	Result
<		`0 < 4`	`True`
<=			
>			
>=			
==		`1 == -1`	`False`
!=		`1 != -1`	`True`

Attention! It's very important not to confuse the operator that checks whether two values are equal (**==, double-equals!**) with the single-equals sign we use to give a name to a certain value (`house = ...`).
Remember: to test two values for equality we use `==`.

Comparing the lengths of two names

In the previous unit we introduced strings as sequences of characters. You can use the `len(...)` function, short for length, to calculate the length of a string.

Program this function so that it produces “Wow!” if the two names passed as arguments have equal length, and “Too bad!” if it is different.

```
def compare(first_name, second_name):
    return
```

Now write it in a different but equivalent way:

```
def compare(first_name, second_name):
    return
```

Lake Lugano

The water level in Lake Lugano is constantly kept under control. We can try to represent it graphically!

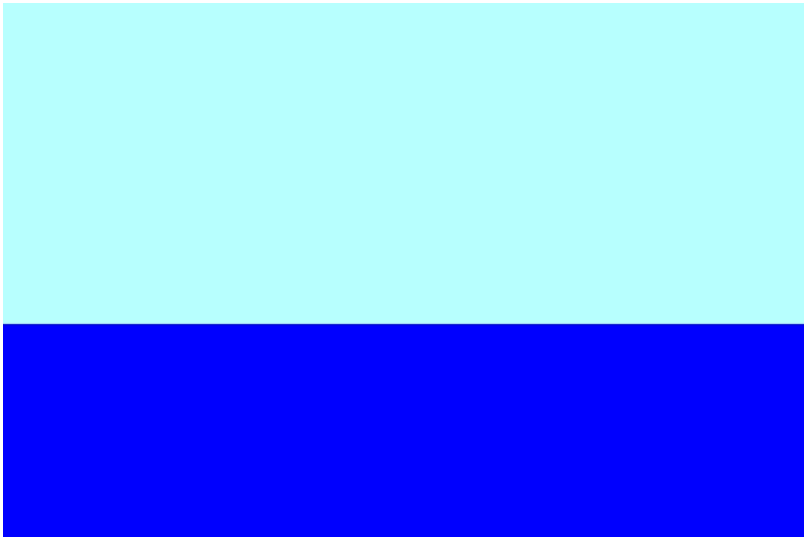
The level of the lake is measured in meters above sea level and is reported to the nearest centimeter. For example, the measurement `270.60` indicates a height of 270 meters and 60 centimeters above sea level.

Write a function that given a height in meters, returns how many centimeters it exceeds 269. For the value 271.10, it must, for example, return 210.

```
def cm_above_269(height_m):
    return

print(cm_above_269(
```

Write a function that produces this graphic depicting the height of the lake in a simplified way. The background must be light blue (RGB values: 183, 254, 255) of size 400 x 300. The height of the lake is represented with a blue rectangle aligned with the bottom edge. Use the function cm_above_269 to know how high the lake should be.



```
def lake(height_m):
    return
```

Try your function on the PyTamaro web site (using values like 271.10, 271.34, ...).

The Federal Environmental Bureau makes level data available to us. We can put all the lake height data day by day into a list, to be visualized later as an animation. Remember `map`?

```
heights = [271.10, 271.15, 271.17, 271.16, 271.10]

show_animation(
```

Try your function on the PyTamaro web site.

When the height exceeds a certain threshold, the situation becomes potentially dangerous and authorities issue an alert to the public.

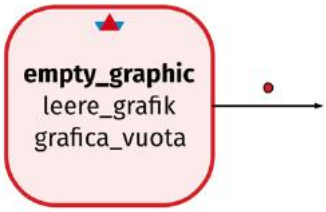
For Lake Lugano, the alert threshold starts at `270.90`.

First create a graphic showing a warning sign:

```
warning_sign =
```

Now we have to program a selection: if the height is higher than the established threshold, we have to add the warning sign to the lake graphic.

And otherwise? Well, we don't have to add anything. PyTamaro offers a function just for this case: `empty_graphic`. Calling this function (which has no parameters) results in an empty graphic. We can combine it with any other graphic and we will always get the other graphic without any modification.



Write this function that decides whether we should show the warning sign or an empty graphic, choosing based on the height value.

```
def warning_graphic(height_m):
    return
```

Now we have to produce the new animation, using `map` again. This time **overlay** the graphic you get from `warning_graphic` on top of the lake graphic. Always use the `heights` list.

```
show_animation(
```

Try your function on the PyTamaro web site (with actual data from Lake Lugano!)