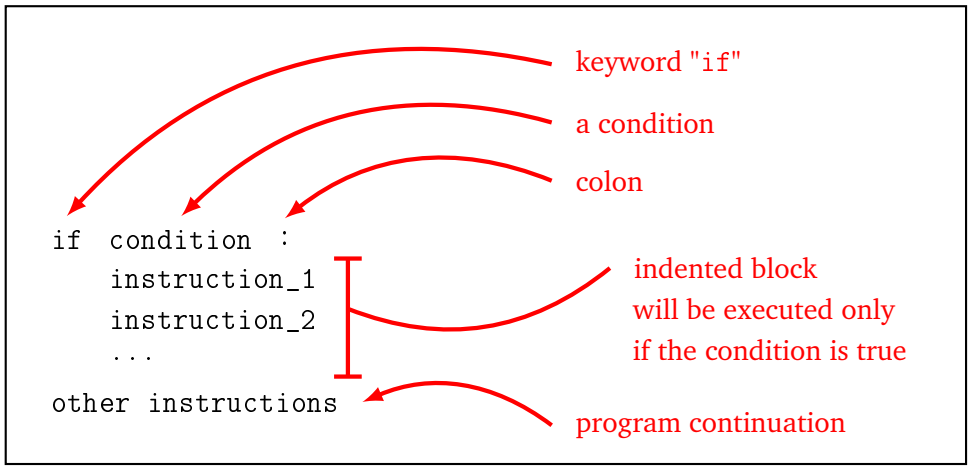
If ... then ...

The computer can react according to a situation. If a condition is met, it acts in a certain way, otherwise it does something else.

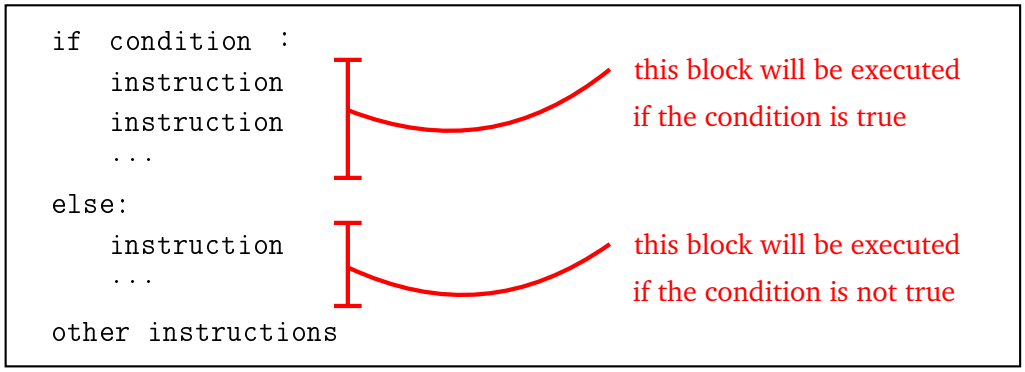
Lesson 1 (If ... then ...).
Here's how to use the "`if`" test with `Python`:



Here is an example, which warns a driver if a variable `speed` is too large.

```
if speed > 110:
    print("Warning, you are driving too fast.")
```

Instructions can also be executed if the condition is not met using the keyword "`else`".



Once again, it is the indentation that delimits the different blocks of instructions. Here is an example that displays the sign of a number `x`.

```
if x >= 0:
    print("The number is positive (or zero).")
else:
    print("The number is negative.")
```

Lesson 2 (Keyboard entry)**.**
To be able to interact with the user, you can ask him to enter text on the keyboard. Here is a small program that asks for the user's first name and age and displays a message like "Hello Kevin" then "You are a minor/adult!" according to age.

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

age_str = input("How old are you? ")
age = int(age_str)

if age >= 18:
    print("You're an adult!")
else:
    print("You're a minor!")
```

Explanations.

- The command `input()` pauses the execution of the program and waits for the user to send a text input (ended by pressing the "Enter" key).
- This command returns a string.
- If you want an integer, you have to convert the string. For example, here `age_str` can be equal to `"17"` (it is not a number but a sequence of characters), while `int(age_str)` is now the integer `17`.
- The reverse operation is also possible, `str()` converts a number into a string. For example `str(17)` returns the string `"17"`; if you set `age = 17`, then `str(age)` also returns `"17"`.

Lesson 3 (The "random" module)**.**
The `random` module generates numbers as if they were randomly drawn.

- Here is the command to place at the beginning of the program to call this module:
  `from random import *`
- The command `randint(a,b)` returns a random integer between a and b.
  For example after the instruction `n = randint(1,6)`, n is a random integer such that $1 \leqslant n \leqslant 6$. If you repeat the instruction `n = randint(1,6)`, n takes a new value. It's like rolling a 6-face dice.
- The command `random()`, without argument, returns a floating number (i.e. a decimal number) between 0 and 1. For example, after the instruction `x = random()`, then x is a floating point number with $0 \leqslant x < 1$.

Activity 1 (Multiplication quiz)**.**

Goal: program a small multiplication tables test.

- Define a variable a, to which you assign a random value between 1 and 12.
- Same thing for a variable b.
- Display the question on the screen: "What is the product $a \times b$?". (Replace a and b by their value!)
- Retrieve the user's answer and transform it into an integer.
- If the answer is correct, display "Well done!", otherwise display "Lost! The correct answer was ...".

Test of equality. To test if two numbers x and y are equal, the instruction is:

```
if x == y:
```

The equality test is written with the double equal symbol "==". For example "`x == 3`" returns "True" if x is equal to 3 and "False" otherwise.
Attention! The instruction "`x = 3`" has nothing to do with testing equality, this instruction stores 3 in the variable x.

Activity 2 (The words of the turtle)**.**

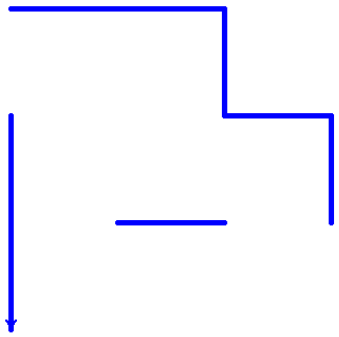
Goal: guide the turtle with a word, each character corresponding to an instruction.

You give the turtle a word, for example **FlFrfflFrfF**, in which each character (read from left to right) corresponds to an instruction that the turtle must execute.

- **F** : go forward of 100 by tracing,
- **f** : go forward of 100 without tracing,
- **l** : turn left by 90 degrees,
- **r** : turn right by 90 degrees.

Example. Here is the drawing that the turtle must trace when

```
word = "FflFlFrFlFFlfFF"
```



Hints. Here's how to scan the letters of a word and test if a letter is the character **F**:

```
for c in word:
    if c == "F":
        instructions...
```

Finally remember that `up()` and `down()` sets the position of the pen.

Lesson 4 (Booleans).

- A ***bolean*** is a type of data that is either equal to the value "True" or the value "False". In `Python` the values are `True` and `False` (with a capital letter).
- We can obtain a boolean as a result of the comparison of two numbers. For example `7 < 4` is equal to `False` (because 7 is not smaller than 4). Check that `print(7 < 4)` displays `False`.
  Here are the main comparisons:
  - **Test of equality:** `a == b`
  - **Strict lower test:** `a < b`
  - **Large lower test:** `a <= b`
  - **Higher test:** `a > b` or `a >= b`
  - **Test of non equality:** `a != b`

  For example `6*7 == 42` is equal to `True`.
-

  ATTENTION! The classic mistake is to confuse "`a = b`" and "`a == b`".
  - **Assignment.** `a = b` puts the content of the variable `b` in the variable `a`.
  - **Test of equality.** `a == b` tests if the contents of `a` and `b` are the same, and is equal to `True` or `False`.

- We can compare something other than numbers. For example, "`char == "A"`" tests if the variable `char` is equal to `"A"`; "`its_raining == True`" tests if the variable `its_raining` is true...
- Booleans are useful in "if ... then ..." tests and in "while ... then ..." loops.
- **Operations between booleans.** If P and Q are two booleans, new booleans can be defined.
  - **Logical and.** "`P and Q`" is true if and only if P and Q are true.
  - **Logical or.** "`P or Q`" is true if and only if P or Q is true.
  - **Negation.** "`not P`" is true if and only if P is false.

  For example "`(2+2 == 2*2) and (5 < 3)`" returns `False`, because even if we have $2+2=2\times2$, the other condition is not satisfied because $5<3$ is wrong.

Activity 3 (Digits of an integer).

Goal: find numbers whose digits verify certain properties.

1. The following program displays all integers from 0 to 99. Understand this program. What do the variables u and t represent?

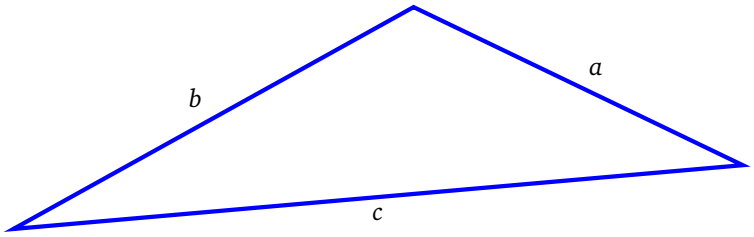
```
for t in range(10):
    for u in range(10):
        n  = 10*t + u
        print(n)
```

2. Find all integers between 0 and 999 that satisfy all the following properties:
   - the integer ends with a 3,
   - the sum of the digits is greater than or equal to 15,
   - the tens digit is even.
3. Modify your previous program to count and display the number of integers that satisfy these properties.

Activity 4 (Triangles)**.**

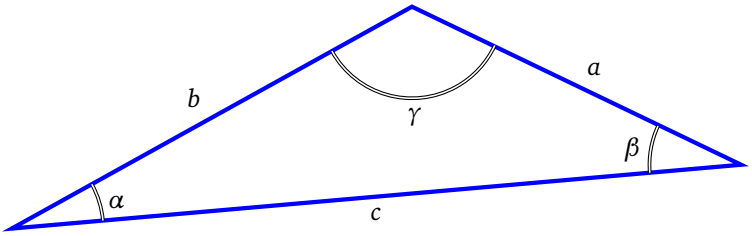
Goal: determine the properties of a triangle from the three lengths of the sides.

We give ourselves the three lengths a, b and c. You will determine the properties of the triangle whose lengths would be a, b, c.



Define three variables a, b and c with integer values and $a \leqslant b \leqslant c$ (or ask the user for three values).

1. **Order.** Ask `Python` to test if the lengths satisfy $a \leqslant b \leqslant c$. Display a sentence for the answer.
2. **Existence.** There is a triangle corresponding to these lengths if and only if:
$$a + b \geqslant c.$$
Ask `Python` to test if this is the case and display the answer.
3. **Right triangle.** Ask `Python` to test if the triangle is a right triangle. (Think of Pythagorean theorem.)
4. **Equilateral triangle.** Test if the triangle is equilateral.
5. **Isosceles triangle.** Test if the triangle is isosceles.
6. **Acute triangle.** Tests if all angles are acute (i.e. less than or equal to 90 degrees).
Hints.
   - The cosine law allows us to calculate an angle according to the lengths:



$$\cos\alpha = \frac{-a^2+b^2+c^2}{2bc}, \qquad \cos\beta = \frac{a^2-b^2+c^2}{2ac}, \qquad \cos\gamma = \frac{a^2+b^2-c^2}{2ab}.$$

   - To test if the angle α is acute just check $\cos\alpha \geqslant 0$ (in the end we never calculate α, but only $\cos\alpha$).

Find examples of lengths a, b, c to illustrate the different properties.

Activity 5 (The mystery number)**.**

Goal: code the classic game when learning to program. The computer chooses a random number, the user must guess this number by following the indications "larger" or "smaller" given by the computer. As this game is quickly boring, we introduce variants where the computer is allowed to lie or cheat!

1. **The classic game.**
   - The computer randomly chooses a mystery number between 0 and 99.

- The player offers an answer.
- The computer replies “the number to find is greater” or “the number to find is smaller” or “bravo, it’s the right number!”.
- The player has seven attempts to find the right answer.

Program this game!

Hints. To leave a `for` loop before the last proposal, you can use the command `break`. Use this when the player finds the right answer.

2. **The computer is lying.**
   To complicate the game, the computer has the right to lie from time to time. For example, about one in four times the computer can give the wrong hint of “larger” or “smaller”.
   Hints. To decide when the computer is lying, each turn, draw a random number between 1 and 4, if it is 4 the computer will lie!
3. **The computer is cheating.**
   Now the computer is cheating (but it no longer lies)! Each turn the computer changes the mystery number a little bit.
   Hints. Each round, draw a random number, between −3 and +3 for example, and add it to the mystery number. (Be careful not to exceed the 0 and 99 limits.)