Conditions: if, then, else

Syntax

All the programs in the first lesson were executed sequentially, line by line. No line could be skipped.

Let's consider the following problems: for the given integer X determine its absolute value. If X>0 then the program should print the value X, otherwise it should print -X. This behavior can't be reached using the sequential program. The program should conditionally select the next step. That's where the conditions help:

```
x = int(input())
if x > 0:
    print(x)
else:
    print(-x)
```

This program uses a conditional statement `if`. After the `if` we put a condition `(x > 0)` following by a colon. After that we put a block of instructions which will be executed only if the condition is true (i.e. evaluates to `True`). This block may be followed by the word `else`, colon and another block of instructions which will be executed only if the condition is false (i.e. evaluates to `False`). Is the case above, the condition if false, so the else-block is executed. Each block should be indented using spaces.

To sum up, the conditional statement in Python has the following syntax:

```
if condition:
    true-block
    several instructions that are executed
    if the condition evaluates to True
else:
    false-block
    several instructions that are executed
    if the condition evaluates to False
```

The `else` keyword with the false-block may be omitted in case nothing should be done if the condition is false. For example, we can replace the variable `x` with its absolute value like this:

```
x = int(input())
if x < 0:
    x = -x
```

```
print(x)
```

In this example the variable `x` is assigned to `-x` only if `x < 0`. In contrast, the instruction `print(x)` is executed every time, because it's not indented, so it doesn't belong to the true-block.

Indentation is a general way in Python to separate blocks of code. All instructions within the same block should be indented in the same way, i.e. they should have the same number of spaces at the beginning of the line. It's recommended to use 4 spaces for indentation.

The indentation is what makes Python different from the most of other language, in which the curly braces `{` and `}`are used to form the blocks.

By the way, there's a builtin-function for absolute value in Python:

```
x = int(input())
print(abs(x))
```

Nested conditions

Any Python instruction may be put into true-blocks and false-block, including another conditional statement. This way we get nested conditions. The blocks of inner conditions are indented using twice more spaces (eg. 8 spaces). Let's see an example. Given the coordinates of the point on the plane, print its quadrant.

```
x = int(input())
y = int(input())
if x > 0:
    if y > 0:
        # x > 0, y > 0
        print("Quadrant I")
    else:
        # x > 0, y <= 0
        print("Quadrant IV")
else:
    if y > 0:
        # x <= 0, y > 0
        print("Quadrant II")
    else:
        # x <= 0, y <= 0
        print("Quadrant III")
```

In this example we use the comments: the explanatory text that has no effect on program execution. This text starts with the hash `#` and lasts till the end of the line.

3. Comparison operators

Usually the condition after `if` has one or more of the following operators:

`<`
: less — the condition is true if left side is less than right side.

`>`
: greater — the condition is true if left side is greater than right side.

`<=`
: less or equal.

`>=`
: greater or equal.

`==`
: equal.

`!=`
: not equal.

For example, the condition `x * x < 1000` means “the value of the expression `x * x` is less than 1000”, and the condition `2 * x != y` means “the doubled value of the variable `x` is not equal to the value of the variable `y`”.

The comparison operators in Python may be chained like this: `a == b == c` or `x <= y >= 10`. It's a rare thing among programming languages.

Bool objects

When we sum two integer objects using the `+` operator, like `2 + 5`, we get a new object: `7`. In the same way, when we compare two integers using the `<` operator, like `2 < 5`, we get a new object: `True`.

```
print(2 < 5)
print(2 > 5)
```

The `True` and `False` objects have a special type called `bool`. As every type name can be used to cast objects into that type, let's see what this cast gives for numbers:

```
print(bool(-10))    # True
print(bool(0))      # False - zero is the only false number
print(bool(10))     # True

print(bool(''))     # False - empty string is the only false string
print(bool('abc'))  # True
```

Logical operators

Sometimes you need to check several conditions at once. For example, you can check if a number `n` is even using the condition `n % 2 == 0` (`n` gives a remainder `0` when dividing by `2`). If you need to check that two numbers `n`and `m` are both even, you should check both `n % 2 == 0` and `m % 2 == 0`. To do that, you join them using an operator `and` (logical AND): `n % 2 == 0 and m % 2 == 0`.

Python has logical AND, logical OR and negation.

Operator `and` is a binary operator which evaluates to `True` if and only if both its left-hand side and right-hand side are `True`.

Operator `or` is a binary operator which evaluates to `True` if at least one of its sides is `True`.

Operator `not` is a unary negation, it's followed by some value. It's evaluated to `True` if that value is `False` and vice versa.

Let's check that at least one of the two numbers ends with 0:

```
a = int(input())
b = int(input())
if a % 10 == 0 or b % 10 == 0:
    print('YES')
else:
    print('NO')
```

Let's check that the number `a` is positive and the number `b` is non-negative:

```
if a > 0 and not (b < 0):
```

Instead of `not (b < 0)` we can write `(b >= 0)`.

if-elif-else

If you have more than two options to tell apart using the conditional operator, you can use `if... elif... else`statement.

Let's show how it works by rewriting the example with point (x,y) on the plane and quadrants from above:

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
    print("Quadrant I")
elif x > 0 and y < 0:
    print("Quadrant IV")
elif y > 0:
    print("Quadrant II")
```

```
else:
    print("Quadrant III")
```

In this case the conditions in `if` and `elif`s are checked one after another until the first true condition is found. Then only the true-block for that condition is being executed. If all the conditions are false, the else-block is being executed, if it's present.

Problem «Minimum of two numbers» (Easy)

Statement

Given two integers, print the smaller value.

Your solution

```
a = int(input())
b = int(input())
if a < b:
    print(a)
else:
    print(b)
```

Problem «Sign function» (Easy)

Statement

For the given integer X print `1` if it's positive, `-1` if it's negative, or `0` if it's equal to zero.

Try to use the cascade `if-elif-else` for it.

Your solution

```
# Read an integer:
X = int(input())
# Print a value:
if X > 0:
    print(1)
elif X < 0:
    print(-1)
else:
    print(0)
```

Problem «Minimum of three numbers» (Easy)

Statement

Given three integers, print the smallest value.

Your solution

```
# Read in intergers
a = int(input())
b = int(input())
c = int(input())
# Print least of three intergers
if a < b and a < c:
    print(a)
elif b < a and b < c:
    print(b)
else:
    print(c)
```

Problem «Equal numbers» (Easy)

Statement

Given three integers, determine how many of them are equal to each other. The program must print one of these numbers: 3 (if all are the same), 2 (if two of them are equal to each other and the third is different) or 0 (if all numbers are different).

Your solution

```
# Read an integers
a = int(input())
b = int(input())
c = int(input())
# how many intergers are equal
if a == b and b == c:
    print(3)
elif (a == b) or (b == c) or (a == c) :
    print(2)
else:
    print(0)
```

Problem «Rook move» (Medium)

Statement

Chess rook moves horizontally or vertically. Given two different cells of the chessboard, determine whether a rook can go from the first cell to the second in one move.
The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output `YES` if a rook can go from the first cell to the second in one move, or `NO` otherwise.

Your solution

```
# Read in col & rows:
c1 = int(input())
r1 = int(input())
c2 = int(input())
r2 = int(input())
#rook moves horizontally or vertically
if (c1 == c2 and (r1 - r2 != 0)#moves vertically
   or
   r1 == r2 and (c1 - c2 != 0)):#moves horizontally
   print("YES")
else:
   print("NO")
```

Suggested solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if x1 == x2 or y1 == y2:
   print('YES')
else:
   print('NO')
```

Problem «Chess board - same color» (Easy)

Statement

Given two cells of a chessboard. If they are painted in one color, print the word `YES`, and if in a different color - `NO`.

The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell.

Your solution

```
# Read in col & rows:
c1 = int(input())
r1 = int(input())
c2 = int(input())
r2 = int(input())
# even sums of rows & col are black and odd sums white
if (c1+r1) % 2 == (c2+r2) % 2 :
    print("YES")
else:
    print("NO")
```

Suggested solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if (x1 + y1 + x2 + y2) % 2 == 0:
    print('YES')
else:
    print('NO')
```

Problem «King move» (Medium)

Statement

Chess king moves horizontally, vertically or diagonally to any adjacent cell. Given two different cells of the chessboard, determine whether a king can go from the first cell to the second in one move.

The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output `YES` if a king can go from the first cell to the second in one move, or `NO` otherwise.

Your solution

```
x1 = int(input())
```

```
y1 = int(input())
x2 = int(input())
y2 = int(input())
if abs(x1 - x2) <= 1 and abs(y1 - y2) <= 1:
    print('YES')
else:
    print('NO')
```

Suggested solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if abs(x1 - x2) <= 1 and abs(y1 - y2) <= 1:
    print('YES')
else:
    print('NO')
```

Problem «Bishop moves» (Hard)

Statement

In chess, the bishop moves diagonally, any number of squares. Given two different squares of the chessboard, determine whether a bishop can go from the first to the second in one move. The program receives as input four numbers from 1 to 8, specifying the column and row numbers of the starting square and the column and row numbers of the ending square. The program should output `YES` if a Bishop can go from the first square to the second in one move, or `NO` otherwise.

Your solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if abs(x1 - x2) == abs(y1 - y2):
    print('YES')
else:
    print('NO')
```

Suggested solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if abs(x1 - x2) == abs(y1 - y2):
    print('YES')
else:
    print('NO')
```

Problem «Queen move» (Hard)

Statement

Chess queen moves horizontally, vertically or diagonally to any number of cells. Given two different cells of the chessboard, determine whether a queen can go from the first cell to the second in one move.

The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output `YES` if a queen can go from the first cell to the second in one move, or `NO` otherwise.

Your solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if abs(x1 - x2) == abs(y1 - y2) or x1 == x2 or y1 == y2:
    print('YES')
else:
    print('NO')
```

Suggested solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
if abs(x1 - x2) == abs(y1 - y2) or x1 == x2 or y1 == y2:
    print('YES')
else:
    print('NO')
```

Problem «Knight move» (Hard)

Statement

Chess knight moves like the letter L. It can move two cells horizontally and one cell vertically, or two cells vertically and one cells horizontally. Given two different cells of the chessboard, determine whether a knight can go from the first cell to the second in one move.

The program receives the input of four numbers from 1 to 8, each specifying the column and row number, first two - for the first cell, and then the last two - for the second cell. The program should output `YES` if a knight can go from the first cell to the second in one move, or `NO` otherwise.

Your solution

```
# Read in col & rows:
c1 = int(input())
r1 = int(input())
c2 = int(input())
r2 = int(input())
#  Chess knight moves like the letter L
if abs(c1 - c2) == 1 and abs(r1 - r2) == 2:
    print("YES")
elif abs(c1 - c2) == 2 and abs(r1 - r2) == 1:
    print("YES")
else:
    print("NO")
```

Suggested solution

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
dx = abs(x1 - x2)
dy = abs(y1 - y2)
if dx == 1 and dy == 2 or dx == 2 and dy == 1:
    print('YES')
else:
    print('NO')
```

Problem «Chocolate bar» (Medium)

Statement

Chocolate bar has the form of a rectangle divided into $n \times m$ portions. Chocolate bar can be split into two rectangular parts by breaking it along a selected straight line on its pattern. Determine whether it is possible to split it so that one of the parts will have exactly k squares.

The program reads three integers: n, m, and k. It should print `YES` or `NO`.

Your solution

```
n = int(input())
m = int(input())
k = int(input())
if k < n * m and ((k % n == 0) or (k % m == 0)):
    print('YES')
else:
    print('NO')
```

Problem «Leap year» (Easy)

Statement

Given the year number. You need to check if this year is a leap year. If it is, print `LEAP`, otherwise print `COMMON`.

The rules in Gregorian calendar are as follows:

- a year is a leap year if its number is exactly divisible by 4 and is not exactly divisible by 100
- a year is always a leap year if its number is exactly divisible by 400

Warning. The words `LEAP` and `COMMON` should be printed all caps.

Your solution

```
# Read a year
year = int(input())
# Print if leap year or common
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print("LEAP")
else:
    print("COMMON")
```