STEP
IT ACADEMY

Programming with python

Lesson # 5

Conditional Statement

CONTENTS

Logical Operators

You are now familiar with all the kinds of conditions that are in Python, but there is one more construct that simplifies the process of writing programs: **the `and` and `or` commands**.

Before considering them, let's talk about conditions. The conditions that we considered earlier were simple. For example, if you get enough points in a game, then go to the next level. This is a simple condition.

If, in order to move to a new level, you need to build a building, capture an enemy camp, and get 150 units of gold ore, this is a **difficult condition**.

To use complex conditions, there are special commands: `and`, `or`, `not`. These commands are **used to combine several conditions:**

- The **`and` command** requires all conditions to be true. In other words, both expressions must be `true`:

```
a = 5
b = 5
if (a>0) and (b>0):
   statement
   ...
```

- The `or` **command** requires one of the conditions to be true. It is executed if at least one of the conditions is `true`.

```
a = 0
b = 5
if (a>0) or (b>0):
    statement
    ...
```

- The `not` **command** requires the condition to be false.

```
a = 5
if (not a==0):
    statement
```

The below table shows how logical operators work.

A	B	A AND B	A OR B	NOT A
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

It is worth paying attention to the fact that there can be a huge number of conditions, but the rules remain the same.

The if-elif Statement

Conditions are important and often used components of any program. However, it is not always possible to write a condition in the form of an `if-else`, so this construct can be slightly improved by adding the `elif` command. Figure 1 shows how this construct works.

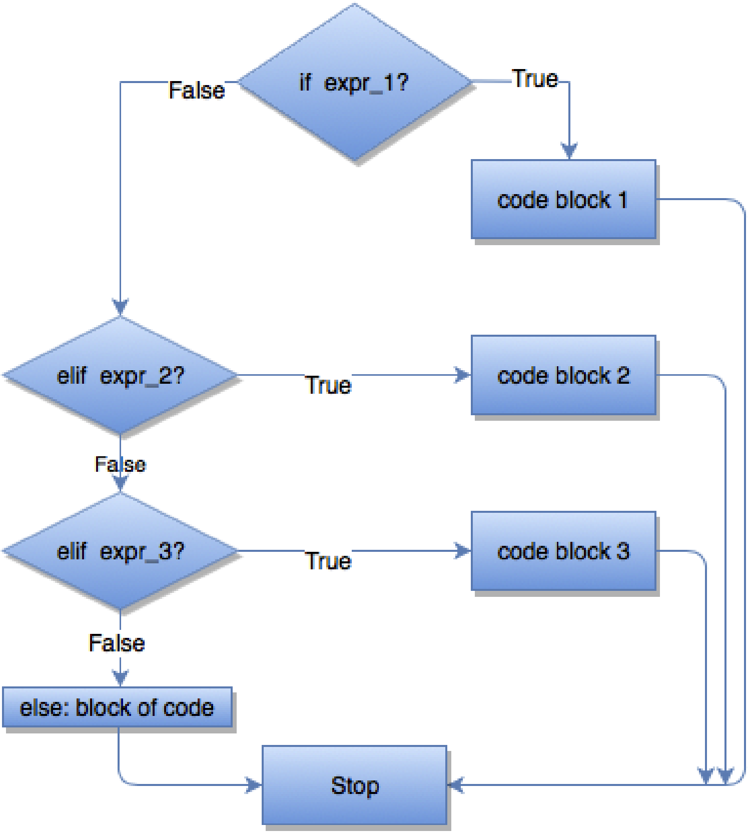

Figure 1

Now let's see what is happening on this scheme.

1. Check the conditions with `if`:

- if it is true, execute the code in block 1;
- if it is false, go to step 2.

2. Check the conditions with `elif`:

- if it is true, execute the code in block 2;
- if it is false, go to step 3.

…

n. The `else` statement terminates this chain. The code in this block will be executed if no condition has been fulfilled before.

Design a Simple PC Assistant

Let’s write a program, a **simple personal assistant** that would analyze the person’s date of birth according to various criteria. It helps you find out the user’s year of birth, age group, and age in seconds (Figure 2).

Figure 2

Take a look at the code:

```
import datetime


print("n---------------------------------------")
print("Enter 1 if you want to know about the year
(365 or 366 days).")
```

```
print("Enter 2 if you want to know about
the age group.")
print("Enter 3 if you want to know about the age
in seconds.")
print("-------------------------------------\n")

birth_day = int(input("Your birth day is xx: "))
birth_month = int(input("Your birth month is xx:
"))
birth_year = int(input("Your birth year is xxxx:
"))
day = int(datetime.date.today().day)
month = int(datetime.date.today().month)
year = int(datetime.date.today().year)
number = int(input("Select what you want: "))


if month > birth_month:
   age = year - birth_year
else:
   age = (year - birth_year) - 1

if (number > 0) and (number < 4) and (age >= 0)
and (age < 130):
   if number == 1:
       if birth_year % 4 != 0:
           print("It is a common year (365 days)")
```

```
        else:
            print("It is a leap year (366 days)")
    elif number == 2:
        print("Your group is ", end="")
        if age < 1:
            print("Baby")
        elif (age >= 1) and (age < 3):
            print("Toddler")
        elif (age >= 3) and (age < 5):
            print("Preschool")
        elif (age >= 5) and (age < 12):
            print("Gradeschooler")
        elif (age >= 12) and (age < 19):
            print("Teen")
        elif age >= 19:
            print("Adult")
    elif number == 3:
        print("Your age: {} year, {} month,
             {} day" .format(age,
             abs(month-birth_month),
             abs(day-birth_day)))
        seconds = ((age * 31536000) + ((month-1) *
                   2592000) + ((day-1) * 86400))
        print("Your age is {} seconds"
              .format(seconds))

else:
    print("Error! Try again.")
```

Let's look at the details of what is going on in this program.

We have already worked with the `datetime` library, and it turned out to be very useful in this program. Using the data on the current year `datetime.date.today().year`, month `datetime.date.today().month`, and day `datetime.date.today().day`, we can calculate the exact age of the person if we know his birth date.

First you need to offer to enter the user's data (day, month, and year of birth):

```
birth_day = int(input("Your birth day is xx: "))
birth_month = int(input("Your birth month is xx: 
"))
birth_year = int(input("Your birth year is xxxx: 
"))
```

Then write the current date in the `day`, `month`, and `year` variables:

```
day = int(datetime.date.today().day)
month = int(datetime.date.today().month)
year = int(datetime.date.today().year)
```

Next our program uses the `if-else` statement. You may ask why do we need it. It's pretty simple, if we do not compare the current month and the month of birth, then we cannot calculate the exact age.

If the program ignores the user's month of birth, it is likely to be wrong in the calculations. To avoid such errors, we need to know the current age:

```
if month > birth_month:
   age = year - birth_year
else:
   age = (year - birth_year) - 1
```

The program uses several nested conditions and the `and` logical operator. This is necessary in order to check the following conditions: whether the action number was entered correctly (there are 3 commands to choose from) and whether the real age was specified:

```
if (number > 0) and (number < 4) and (age >= 0)
and (age < 130):
```

In the first `if`, we compare the number entered by the user (1, 2, or 3) to 1. If the result is `true`, then the program calculates whether this was a leap year. If the result of division is not a multiple of 4, i. e. not divisible by 4 without remainder, then it had 365 days:

```
if number == 1:
       if birth_year % 4 != 0:
           print("It is a common year (365 days)")
       else:
           print("It is a leap year (366 days)")
```

The most unusual in this code is the definition of age, up to the number of months and days, as well as the calculation of age in seconds. Here we need the basic knowledge of mathematics.

Since one year is 31536000 seconds, one month is 2592000 seconds, and one day is 86400 seconds, you should use this code:

```
elif number == 3:
       print("Your age: {} year, {} month,
       {} day" .format(age,
       abs(month-birth_month),
       abs(day-birth_day)))
       seconds = ((age * 31536000) + ((month-1) *
       2592000) + ((day-1) * 86400))
       print("Your age is {} seconds"
             .format(seconds))
```

Writing this program will allow you not only to use the knowledge gained in this lesson but also consolidate the information that you learned in the previous lessons.

STEP
IT ACADEMY

Lesson # 5
Conditional Statement