Programming Fundamentals

Conditionals and selection

João Correia Lopes

INESC TEC, FEUP

04 October 2018

GOALS

By the end of this class, the student should be able to:

- Describe conditionals and selection
- Describe Boolean values, logical operators, and expressions
- Describe the use of if-then-else blocks for conditional execution

BIBLIOGRAPHY

- Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers, How to Think Like a Computer Scientist — Learning with Python 3, 2018 (Section 3.2) [PDF]
- Brad Miller and David Ranum, Learning with Python: Interactive Edition. Based on material by Jeffrey Elkner, Allen B. Downey, and Chris Meyers (Chapter 7) [HTML]

TIPS

- There's no slides: we use a script and some illustrations in the class. That is NOT a replacement for **reading the bibliography** listed in the *class sheet*
- "Students are responsible for anything that transpires during a class—therefore **if you're not in a class**, you should get notes from someone else (not the instructor)"—David Mayer
- The best thing to do is to **read carefully** and **understand** the documentation published in the Content wiki (or else **ask** in the class)
- We will be using **Moodle** as the primary means of communication

CONTENTS

BOOLEAN VALUES AND EXPRESSIONS

- Programs get really interesting when we can test conditions and change the program behaviour
- A *Boolean* value is either true or false
- In Python, the two Boolean values are `True` and `False` and the type is bool
- A Boolean expression is an expression that evaluates to produce a result which is a Boolean value
- For example, the operator == tests if two values are equal

Python

```python
print(True)
print(type(True))
print(type(False))

print(type("True"))
#type(true)

print(5 == (3 + 2))

x = 5
print(x > 0 and x < 1)

n = 25
print(n % 2 == 0 or n % 3 == 0)

age = 19
old_enough_to_get_driving_licence = age >= 18
print(old_enough_to_get_driving_licence)
```

⇒ https://github.com/fpro-admin/lectures/blob/master/04/booleans.py

Comparison operators

```
x == y                    # Produce True if ... x is equal to y
x != y                    # ... x is not equal to y
x > y                     # ... x is greater than y
x < y                     # ... x is less than y
x >= y                    # ... x is greater than or equal to y
x <= y                    # ... x is less than or equal to y
```

- There are three logical operators, `and`, `or`, and `not`
- to build more complex Boolean expressions from simpler Boolean expressions
- The semantics (meaning) of these operators is similar to their meaning in English
- The expression on the left of the `or` operator is evaluated first:
  - if the result is True, Python does not (and need not) evaluate the expression on the right
  - this is called *short-circuit evaluation*
- Similarly, for the `and` operator:
  - if the expression on the left yields False, Python does not evaluate the expression on the right

TRUTH TABLE: AND

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

Truth Table: or

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

Truth Table: not

a	not a
False	True
True	False

Precedence of Operators

Level	Category	Operators
7(high)	exponent	**
6	multiplication	*,/,//,%
5	addition	+,-
4	relational	==,!=,<=,>=,>,<
3	logical	not
2	logical	and
1(low)	logical	or

BOOLEAN ALGEBRA

- A set of rules for simplifying and rearranging expressions is called an *algebra*
- The *Boolean algebra* provides rules for working with Boolean values

Boolean algebra: and

```
x and False == False
False and x == False
y and x == x and y
x and True == x
True and x == x
x and x == x
```

BOOLEAN ALGEBRA: OR

```
x or False == x
False or x == x
y or x == x or y
x or True == True
True or x == True
x or x == x
```

BOOLEAN ALGEBRA: NOT

```
    not (not x) == x
```

CONDITIONAL STATEMENTS: `IF`

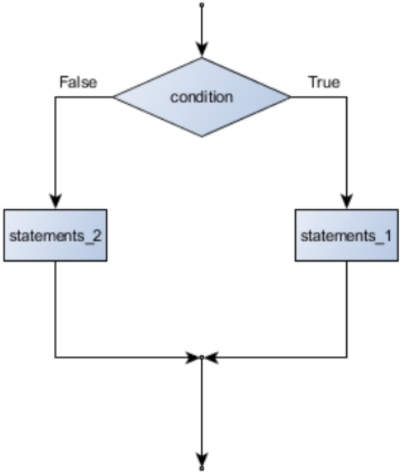
- Conditional statements give us the ability to check conditions and change the behavior of the program accordingly
- The simplest form is the *if statement*
- The Boolean expression after the if statement is called the condition

```
x = 15

if x % 2 == 0:
    print(x, "is even")
else:
    print(x, "is odd")
```

If statement with an else clause

```
if <BOOLEAN EXPRESSION>:
    <STATEMENTS_1>
else:
    <STATEMENTS_2>
```



Blocks and indentation

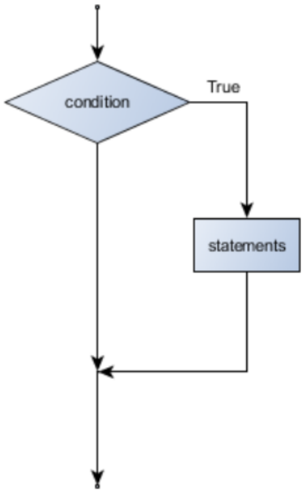
- The indented statements that follow are called a **block**
- The first unindented statement marks the end of the block
- There is no limit on the number of statements that can appear under the two clauses of an if statement
- but there has to be at least one statement in each block
- Occasionally, it is useful to have a section with no statements (usually as a place keeper, or scaffolding, for code we haven't written yet)

```
if True:    # This is always True,
    pass    # so this is always executed, but it does nothing
else:
    pass    # And this is never executed
```

⇒ https://github.com/fpro-admin/lectures/blob/master/04/selections.py

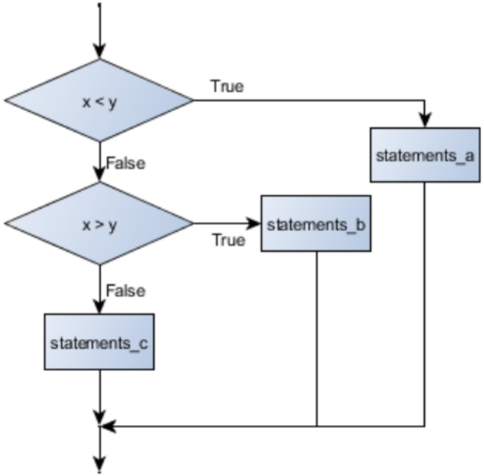
If statement with no else clause: Unary Selection

```
if <BOOLEAN EXPRESSION>:
    <STATEMENTS>
```



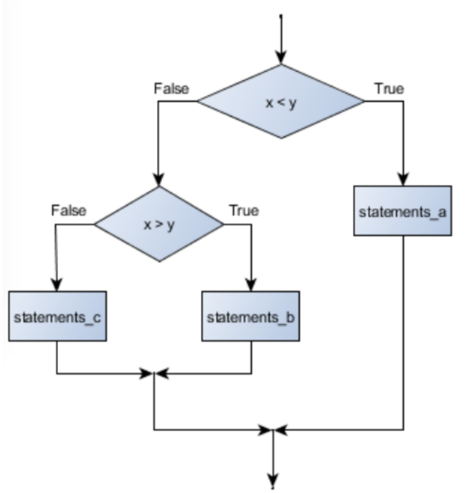
Chained conditionals

```
if <BOOLEAN EXPRESSION_1>:
    <STATEMENTS_A>
elif <BOOLEAN EXPRESSION_2>:
    <STATEMENTS_A>
else:
    <STATEMENTS_C>
```



Nested conditionals

```
if <BOOLEAN EXPRESSION_1>:
    <STATEMENTS_A>
else:
    if <BOOLEAN EXPRESSION_2>:
        <STATEMENTS_B>
    else:
        <STATEMENTS_C>
```



Logical opposites

operator	logical opposite
==	!=
!=	==
<	>=
<=	>
>	<=
>=	<

Exercises

- Moodle activity at: LE04: Conditionals