



universidade de aveiro
departamento de electrónica,
telecomunicações e informática



Fundamentos de Programação

António J. R. Neves
João Rodrigues

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

an@ua.pt / jmr@ua.pt
<http://elearning.ua.pt/>

Fundamentos de Programação 2017/2018





universidade de aveiro
departamento de electrónica,
telecomunicações e informática

Summary

- while statement
- for statement
- range function

Fundamentos de Programação 2017/2018





universidade de aveiro
departamento de electrónica,
telecomunicações e informática

The while statement (1)

- Repeating identical or similar tasks without making errors is something that computers do well.
- Because iteration is so common, *Python* provides several language features to make it easier. One is the `while` statement.
- A `while` loop statement repeatedly executes a target statement as long as a given condition is true.

```
while expression:
    statement(s)
```
- Statement(s) may be a single statement or a block of statements. The condition may be any expression, and `True` is any non-zero value. The loop iterates while the condition is true.
- When the condition becomes false, program control passes to the line immediately following the loop.

Fundamentos de Programação 2017/2018



universidade de aveiro
departamento de electrónica,
telecomunicações e informática

The while statement (2)

- Example:

```
n = 10
while n > 0:
    print(n)
    n -= 1
```
- *Python* supports to have an `else` statement associated with a loop statement.

```
count = 0
while count < 5:
    print(count, " is less than 5")
    count += 1
else:
    print(count, " is not less than 5")
```

Fundamentos de Programação 2017/2018

The while statement (3)



- The body of the loop should change the value of one or more variables so that eventually the condition becomes false and the loop terminates. Otherwise the loop will repeat forever, which is called an infinite loop.
- Sometimes only in the half way through the body it is possible to decide if the cycle should stop. In that case you can use the `break` statement to jump out of the loop.

```
while True:
    line = input('some text: ')
    if line == 'done':
        break
    print(line)
```

Fundamentos de Programação 2017/2018

range function



- The built-in function `range` is a function used to iterate over a sequence of numbers. It generates an iterator of arithmetic progressions.
- The `range` function is generally used to iterate in `for` loops.
- It has two sets of parameters, as follows:
 - `range(stop)`
 - `range(start, stop, step)`
- All parameters must be integers.
- All parameters can be positive or negative.
- `range` (and *Python* in general) is 0-index based, meaning that indexes start at 0. The last integer generated by `range` is up to, but not including, `stop`.

Fundamentos de Programação 2017/2018

for statement (1)



- Another loop mechanism is the `for` statement.
- It has the ability to iterate over the items of any sequence, such as a list or a string.

```
for iterating_var in sequence:
    statements(s)
```

- If a sequence contains an expression list, it is evaluated first.
- Then, the first item in the sequence is assigned to the iterating variable `iterating_var`.
- Next, the statements block is executed. Each item in the list is assigned to `iterating_var`, and the statement(s) block is executed until the entire sequence is exhausted.

Fundamentos de Programação 2017/2018

for statement (2)



- Example:

```
for i in range(4):
    print('Hello!')
```

- The following example illustrates the combination of an `else` statement with a `for` statement that searches for prime numbers from 10 through 20.

```
for num in range(10,20): #to iterate between 10 to 20
    for i in range(2,num): #to iterate between 2 and the number
        if num%i == 0:     #to determine the first factor
            break          #to move to the next number, the #first FOR
    else:                  # else part of the loop
        print(num, 'is a prime number')
```

Fundamentos de Programação 2017/2018



Loop Control Statements



universidade de aveiro
departamento de electrónica,
telecomunicações e informática

- Loop control statements change the execution from its normal sequence (`break`, `continue`, `pass`).
- `break` terminates the loop statement and transfers execution to the statement immediately following the loop.
- The `continue` statement returns the control to the beginning of the current loop. When encountered, the loop starts next iteration without executing the remaining statements in the current iteration.
- `pass` is used when a statement is required syntactically but nothing is needed to be executed. Nothing happens when it is executed. The `pass` statement is also useful in places where the code will eventually go, but has not been written yet