

# Statistical Programming with Python

Pepe García

2020-04-20

# Statistical Programming with Python

Functions + Conditionals

# Plan for today

Boolean expressions

Functions

Conditional execution

# Functions

Functions are sequences of instructions that we store to be executed later

# Calling functions

The syntax for calling functions is the following:

```
function_name(parameter1, parameter2)
```

# Calling functions

# Declaring functions

We can declare our own functions using the `def` keyword with the following syntax:

```
def function_name(parameter1, parameter2):  
    #function body
```

# Declaring functions

When creating a function we need to indent the body. Then we will need to de-indent it to denote the end of the body

```
def function_name(parameter1, parameter2):  
    #function body
```



# Creating functions

# Returning values from functions

Functions in Python can return values after doing all the operations they perform.

# Returning values from functions

# Functions can have parameters

Parameters are values that are injected to the function body when we call it

# Functions can have parameters

# Boolean operations

We're going to learn two kinds of operators that operate on booleans: Comparison and logical operators.

Boolean operations are useful for conditional execution.

# Comparison operators

name	description\
<code>x == y</code>	x is equal to y
<code>x != y</code>	x is not equal to y
<code>x &gt; y</code>	x is greater than y
<code>x &lt; y</code>	x is lesser than y
<code>x &gt;= y</code>	x is greater than or equal than y
<code>x &lt;= y</code>	x is lesser than or equal than y
<code>x is y</code>	x is the same as y
<code>x is not y</code>	x is not the same as y

# Comparison operators



# Logical operators

name	description
x and y	returns True if x and y are true
x or y	returns True if either x or y are true
not x	negates x

# Logical operators

# Conditional execution

Almost all useful programs need to be able to check conditions and change its behaviour accordingly. That's what conditional execution provides.

# Conditional execution

# If statement

the if statement is the tool we use for conditional execution in Python

```
if <condition>:  
    <body>
```

# If statement

# Else clause

The else clause is executed when the condition is evaluated to false:

```
if <condition>:  
    <block>  
else:  
    <block>
```

# Else clause



# Elif clause

Elif clauses are used when there are more possibilities:

```
if <condition>:  
    <block>  
elif <condition>:  
    <block>  
else:  
    <block>
```

# Elif clause

# Recap

Create functions with `def`. Return to produce a value at the end

Combine comparison & logical operators to check the conditions you need

Use `if`, `else`, `elif` for conditional execution

# Homework

1. Create a function `~weekly_commute_time~` that asks the user their daily commute time and returns their weekly time spent commuting.
2. What do the following expressions return?
  1. `~True or 11 > 34~`
  2. `~False and (1 == 1)~`
  3. `~(77 // 11) > 6 and False~`
3. Create a function `~area_triangle~` that takes the base and height of a triangle and returns its area
4. Create function `~area_triangle_rectangle~` that takes the base, height, and the kind of shape and calculates its area. It should work for both triangles and rectangles.
5. Create a function `~im_in_love~` that takes a weekday number (from

# Optional work

1. Create a function that solves quadratic equations.
2. Create a function to calculate the area of (regular) pentagons
3. Create a function to calculate the area of squares
4. Create a function that receives the side, and a polygon name (either pentagon or square), and calculates its area.