Python for Beginners

Pepe García

2020-04-20

Plan for today

Plan for today

Functions

for not repeating ourselves ;)

Plan for today

Functions

for not repeating ourselves ;)

Boolean operators

operators to produce or combine boolean values

Plan for today

Functions

for not repeating ourselves ;)

Boolean operators

operators to produce or combine boolean values

Conditional execution

to make our programs branch

Functions

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

Functions

Calling functions

The syntax for calling functions is the following:

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

Functions

Calling functions

The syntax for calling functions is the following:

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

Demo

Let's do a small demo with the functions we already know

Functions

Declaring functions

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

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

Functions

Declaring functions

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

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

When creating a function we need to indent the body to tell Python what piece of code we want to include inside the function.

Functions

Declaring functions

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

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

When creating a function we need to indent the body to tell Python what piece of code we want to include inside the function.

Demo

Functions

Returning values from functions

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

Functions

Returning values from functions

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

Demo

Functions

Function Parameters

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

Functions

Function Parameters

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

Demo

Checkpoint

Regarding functions, we've seen:

- Functions

Checkpoint

Regarding functions, we've seen:

- Functions
- Calling them

Checkpoint

Regarding functions, we've seen:

- Functions
- Calling them
- Declaring them

Checkpoint

Regarding functions, we've seen:

- Functions
- Calling them
- Declaring them
- Returning values from them

Checkpoint

Regarding functions, we've seen:

- Functions
- Calling them
- Declaring them
- Returning values from them
- Parameters

Checkpoint

Regarding functions, we've seen:

- Functions
- Calling them
- Declaring them
- Returning values from them
- Parameters
- **Questions?**

Boolean operations

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

Boolean operations are useful for conditional execution.

Comparision operators

name	description
`x == y`	x is equal to y
`x != y`	x is not equal to y
`x > y`	x is greater than y
`x < y`	x is lesser than y
`x >= y`	x is greater than or equal than y
`x <= y`	x is lesser than or equal than y

Comparision operators

Demo

- Are two strings the same?
- Are two boolean values different?
- Is this number greater than or equal that other one?

Logical operators

We use logical operators to combine boolean values. They are the operators with the lowest precedence, any other expression will be evaluated before them.

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

We use logical operators to combine boolean values. They are the operators with the lowest precedence, any other expression will be evaluated before them.

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

Demo

Conditional execution

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

`if` statement

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

if statement

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

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

if statement

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

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

Demo

- What type will the condition in our if statement have?

if statement

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

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

Demo

- What type will the condition in our if statement have?
- How can we create a if statement that always executes its body?

if statement

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

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

Demo

- What type will the condition in our if statement have?
- How can we create a if statement that always executes its body?
- What about one that never does it?

Else clause

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

Else clause

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

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

Else clause

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

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

Demo

- Check if a user can drive

Else clause

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

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

Demo

- Check if a user can drive
- Tell him to wait some time if they can't

Elif clause

Elif clauses are used when there are more possibilities:

Elif clause

Elif clauses are used when there are more possibilities:

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

Elif clause

Elif clauses are used when there are more possibilities:

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

Demo

- Check if a user can drive

Elif clause

Elif clauses are used when there are more possibilities:

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

Demo

- Check if a user can drive
- Check if they're accompanied by an adult

Elif clause

Elif clauses are used when there are more possibilities:

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

Demo

- Check if a user can drive
- Check if they're accompanied by an adult
- Tell them to wait otherwise

Exercise time!

Let's do an exercise. We have to create a function that can calculate the area of either a triangle or a rectangle.

Let's spend 5 mins trying to solve it and we'll do that afterwards together.

Recap

Recap

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

Recap

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

Combine comparision & logical operators to check the conditions you need

Recap

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

Combine comparision & logical operators to check the conditions you need

Use if, else, elif for conditional execution

Exercises

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?
   - `True or 11 > 34`
   - `False and (1 == 1)`
   - `(77 // 11) > 6 and False`
3. Create a function `area_triangle` that takes the base and height of a triangle and returns its area

(cont)

Exercises (Cont)

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 monday to friday), and returns how that weekday is (according to The Cure!):

```
I don't care if Monday's blue
Tuesday's grey and Wednesday too
Thursday I don't care about you
It's Friday, I'm in love
```