Programming Thinking

Session 4

Pepe García jgarciah@faculty.ie.edu

SCHOOL OF SCIENCE & TECHNOLOGY

Plan for today

Plan for today

Boolean operators

operators to produce or combine **bool** values

Plan for today

Boolean operators

operators to produce or combine **bool** values

Conditional execution

to make our programs branch

Plan for today

Boolean operators

operators to produce or combine **bool** values

Conditional execution

to make our programs branch

Functions

for not repeating ourselves ;)

SCHOOL OF SCIENCE & TECHNOLOGY

Boolean operations

We're going to learn two kinds of operators that operate on **bool** values:

- comparision operators
- logical operators

Boolean operations are useful for conditional execution.

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

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

Demo

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

Logical operators

We use logical operators to combine **bool** 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**

SCHOOL OF SCIENCE & TECHNOLOGY

Logical operators

We use logical operators to combine **bool** 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

Learn about truth tables

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Any questions or comments about...

- Comparision operators
- logical operators

SCHOOL OF SCIENCE & TECHNOLOGY

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>
```

SCHOOL OF SCIENCE & TECHNOLOGY

`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?

SCHOOL OF SCIENCE & TECHNOLOGY

`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?

SCHOOL OF SCIENCE & TECHNOLOGY

`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?

SCHOOL OF SCIENCE & TECHNOLOGY

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>
```

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

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>
```

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Any questions or comments about conditional execution?

Functions

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

Functions

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

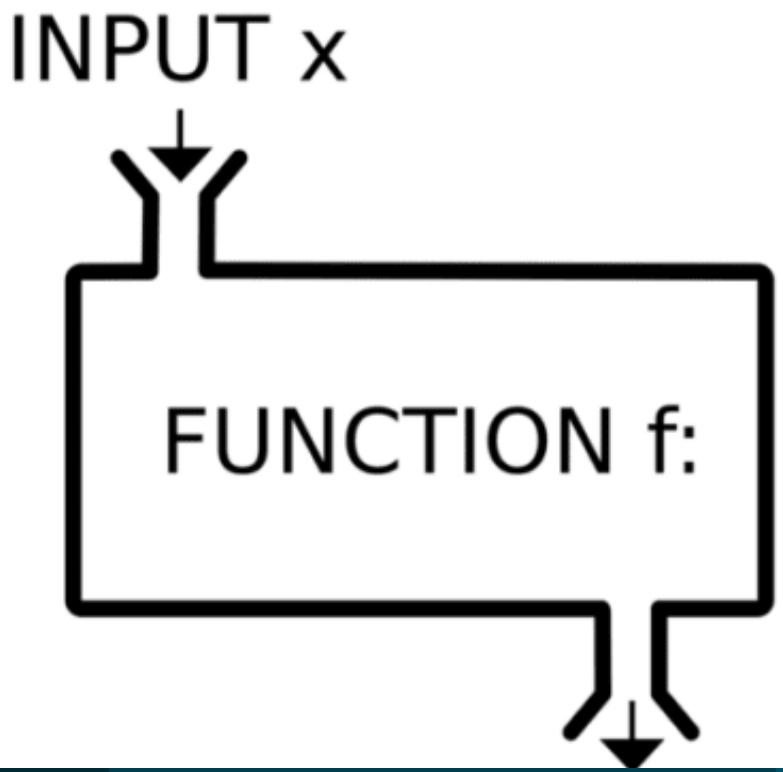
Functions receive **input** as parameters, process the input, and produce **output** as return values.

SCHOOL OF SCIENCE & TECHNOLOGY

Functions

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

Functions receive **input** as parameters, process the input, and produce **output** as return values.



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
```

SCHOOL OF SCIENCE & TECHNOLOGY

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.

SCHOOL OF SCIENCE & TECHNOLOGY

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

Illustrate why indentation is needed.

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Regarding functions, we've seen:

- Functions

Checkpoint

Regarding functions, we've seen:

- Functions
- Calling them

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Regarding functions, we've seen:

- Functions
- Calling them
- Declaring them

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Regarding functions, we've seen:

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

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Regarding functions, we've seen:

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

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Regarding functions, we've seen:

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

SCHOOL OF SCIENCE & TECHNOLOGY

Practice time!

Let's do some practice. We have to create a function `calculate_area_triangle_rectangle` that can calculate the area of either a triangle or a rectangle.

Let's spend 5 mins trying to solve it individually 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

SCHOOL OF SCIENCE & TECHNOLOGY

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

SCHOOL OF SCIENCE & TECHNOLOGY

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 `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
```

SCHOOL OF SCIENCE & TECHNOLOGY

Recommended read

https://automatetheboringstuff.com/ is a great resource for learning how to apply Python to day to day tasks.

It's a long book, don't try to read it cover to cover but instead pick up the chapters that catch your eye from the index.

You can start reading the chapters on basics, functions, and control flow.

SCHOOL OF SCIENCE & TECHNOLOGY