UNIVERSITÄT BIELEFELD
Faculty of Technology

Programming

Programming & Python Basics

Daniel Dörr

Faculty of Technology, Bielefeld University

Recap

Arithmetic in Python

Numeric types:

- Integer: `int` `42`
- Real valued numbers: `float` `42.0`
- Complex numbers: `complex` `42+0j`

Operators

- Addition and subtraction `+ -`
- Multiplication and division `* / // %`
- Exponentiation `**`

Variables

Variable assignment

- `a = 42`
- `b = a - 6.0`

`type(«name of the variable»)`: returns type of variable

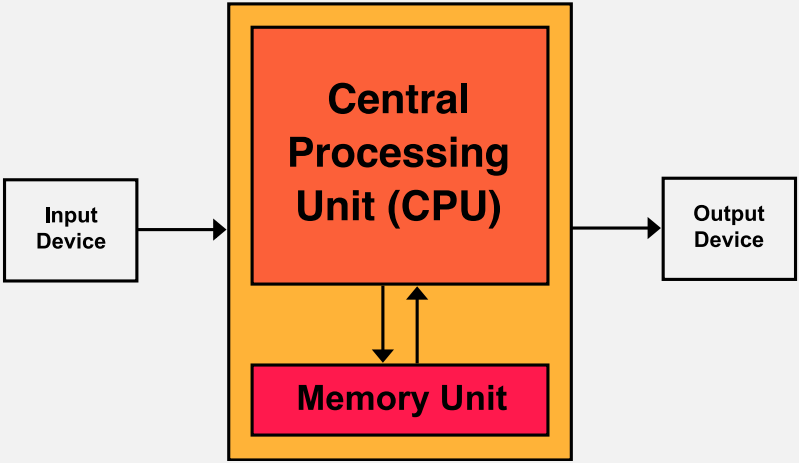
Programming Basics

Data Types & Mutability

Evaluation Order

Conditions & Comparisons

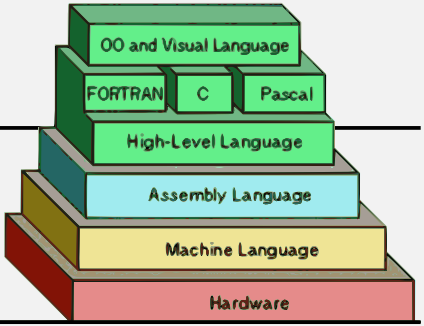
Computer architecture



Python compiler: reference implementation

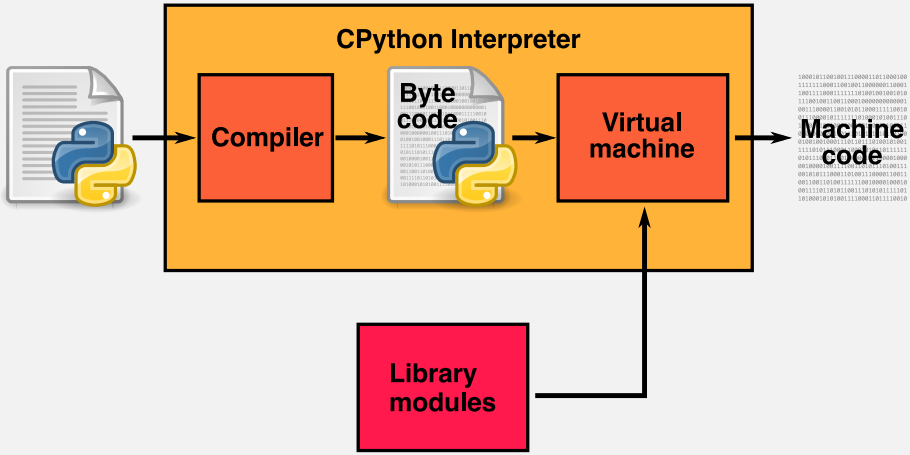
High level language
- Easy to understand

Low level language
- Machine code (processor instructions and binary code)



Source: https://thebittheories.com

Python compiler: reference implementation



Quiz

True or *false*?

- “The three parts of the Von Neumann computer architecture are: Processor, RAM & Hard Disk.”
- “The CPython interpreter converts Python code to the language that the computer’s hardware understands.”

When you try to run Python scripts, a multi-step process begins. In this process the interpreter performs three steps:

1. Ship off the code for execution.
2. Process the statements of your script in a sequential fashion.
3. Compile the source code to an intermediate format known as bytecode.

Identify the correct order of these steps:

1 → 3 → 2 2 → 3 → 1 1 → 2 → 3 3 → 2 → 1

source: https://realpython.com/quizzes

Quiz

True or *false*?

- “The three parts of the Von Neumann computer architecture are: Processor, RAM & Hard Disk.” false
- “The CPython interpreter converts Python code to the language that the computer’s hardware understands.” true

When you try to run Python scripts, a multi-step process begins. In this process the interpreter performs three steps:

1. Ship off the code for execution.
2. Process the statements of your script in a sequential fashion.
3. Compile the source code to an intermediate format known as bytecode.

Identify the correct order of these steps:

1 → 3 → 2 2 → 3 → 1 ✔ 1 → 2 → 3 3 → 2 → 1

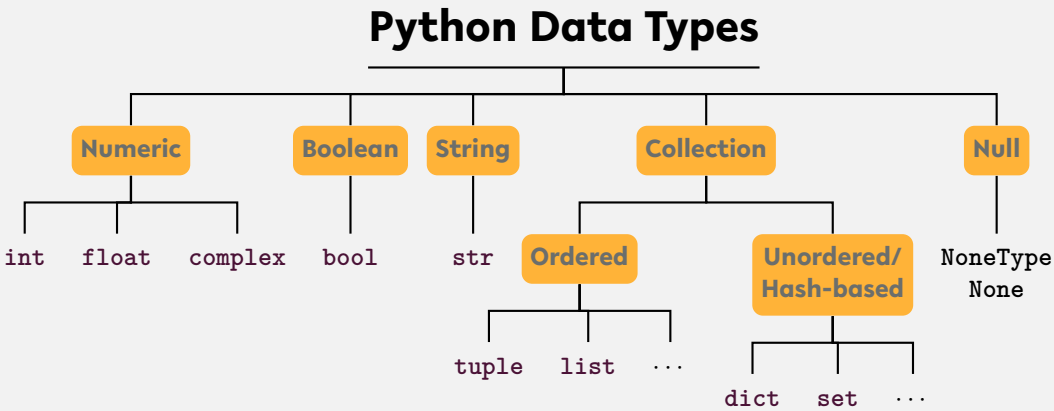
source: https://realpython.com/quizzes

Programming Basics

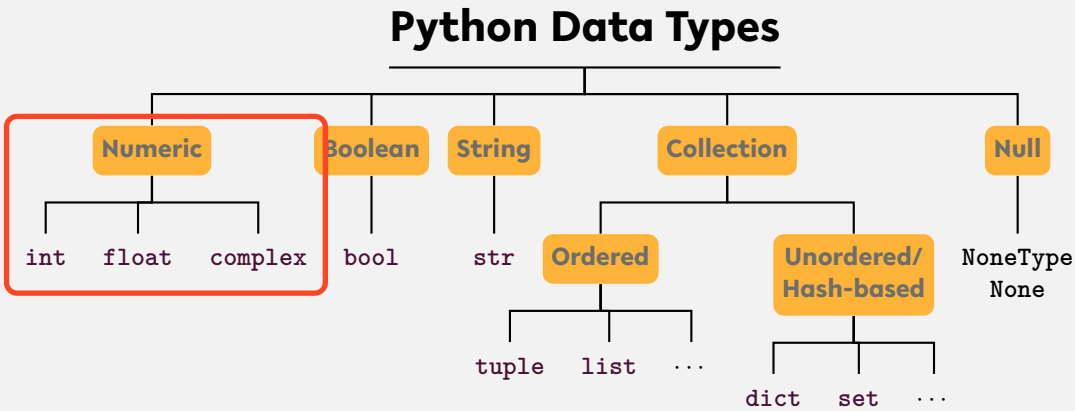
Data Types & Mutability

Evaluation Order

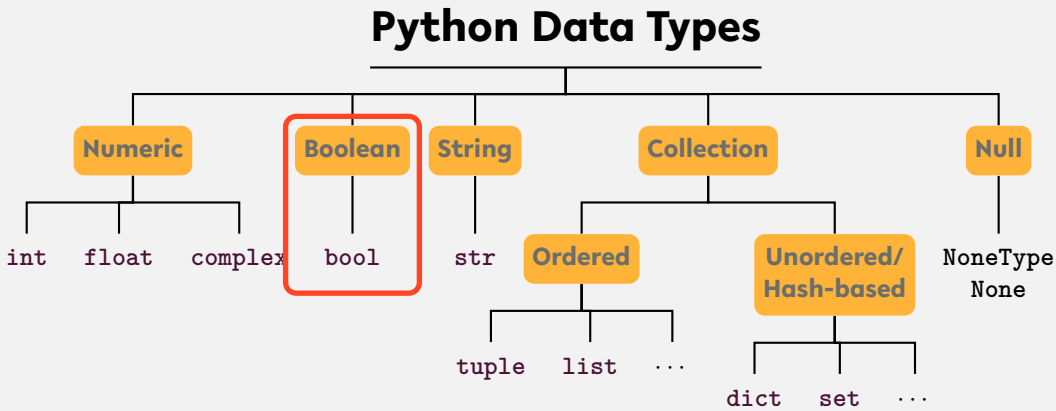
Conditions & Comparisons



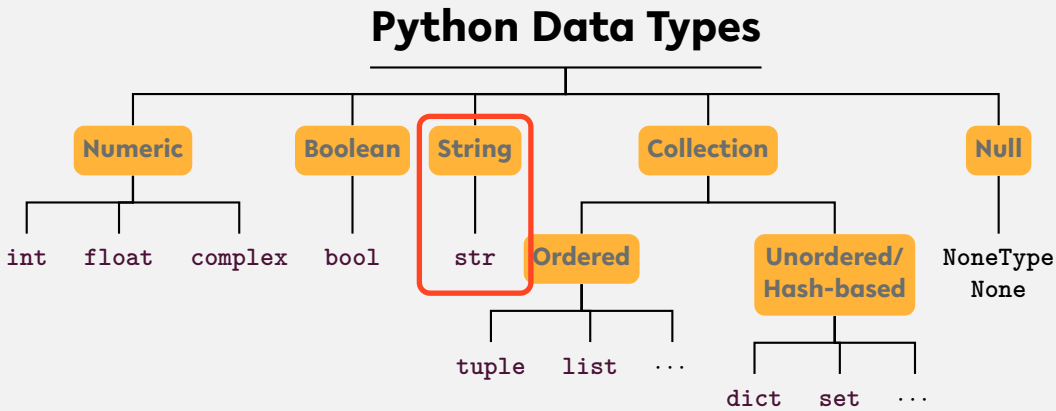
... and user-defined types



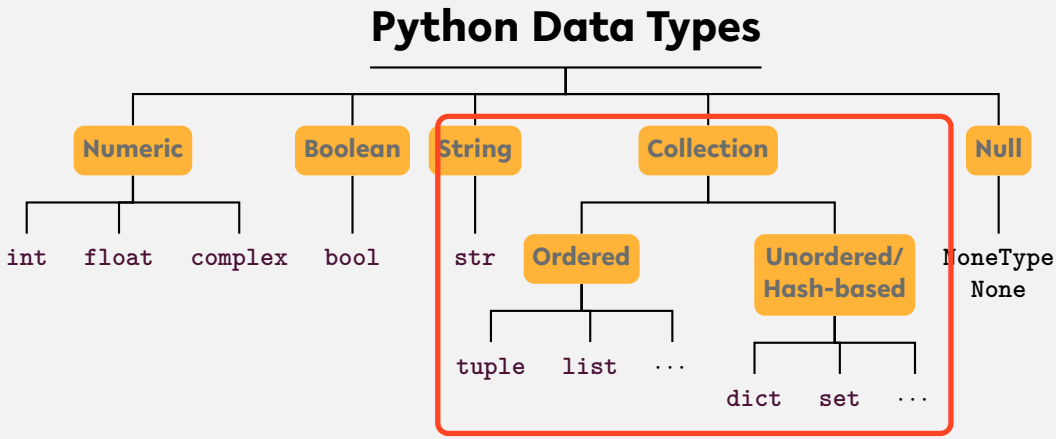
... and user-defined types



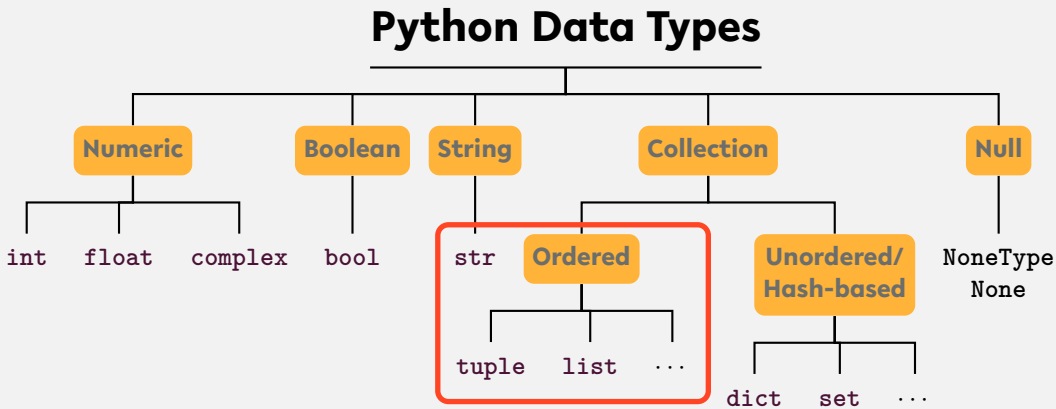
... and user-defined types



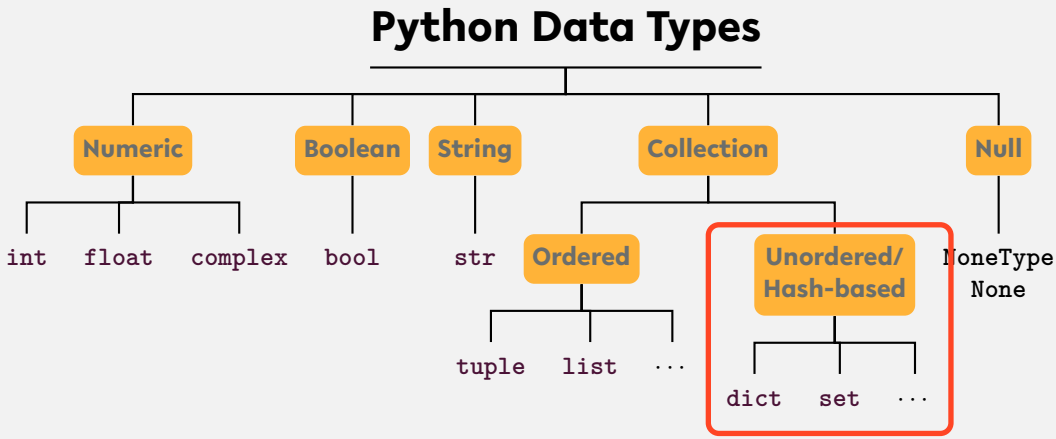
... and user-defined types



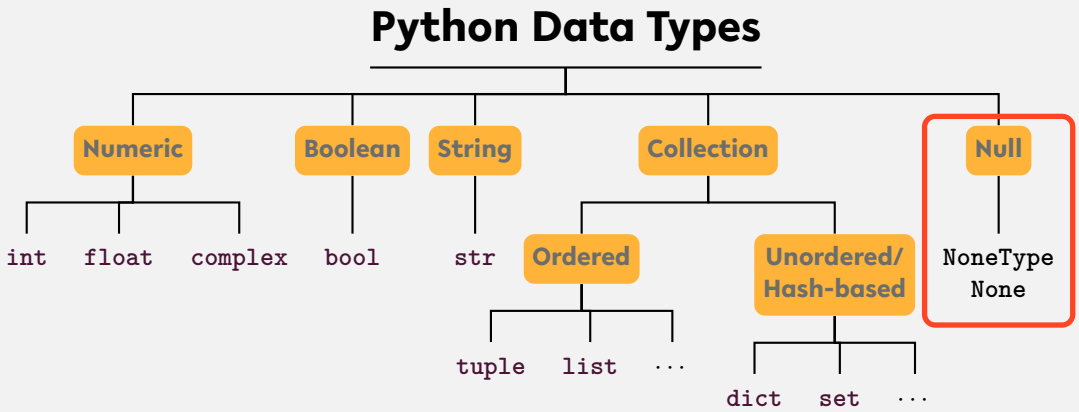
... and user-defined types



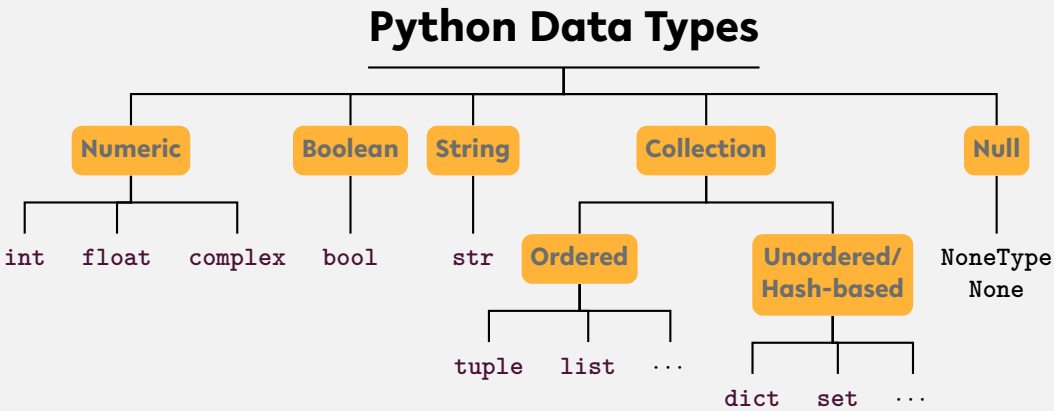
... and user-defined types



... and user-defined types

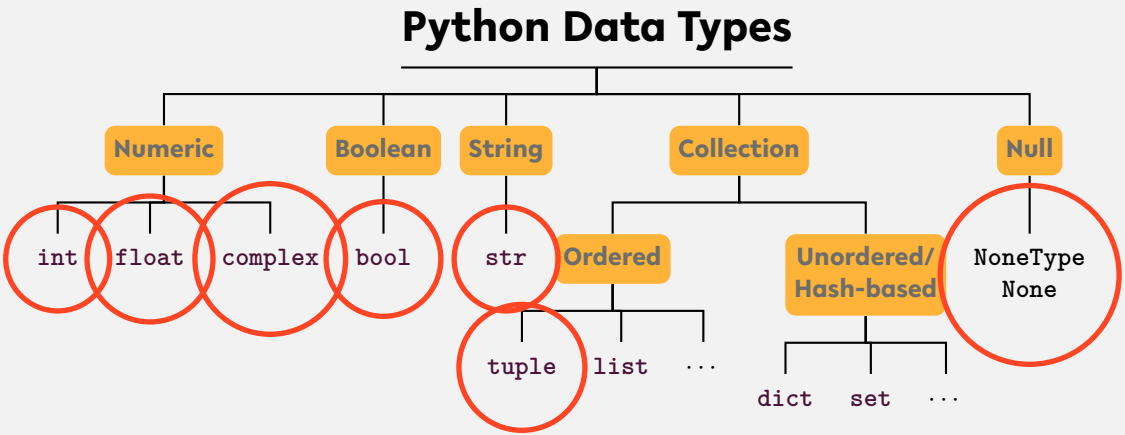


... and user-defined types



... and user-defined types

We differentiate between **type** and **instance**!



... and user-defined types

Instances of certain types are **immutable** , i.e., **cannot be changed after creation**

Memory address: `id`

- Every instance has a unique address in memory
- `id(x)`: memory address of instance of `x`
- `x`, `y` reference the same instance if and only if `x` is equal to `y`.

Memory address: `id`

- Every instance has a unique address in memory
- `id(x)`: memory address of instance of `x`
- `x`, `y` reference the same instance if and only if `x` is equal to `y`.
- *Fun fact: CPython binds integers from -5 to 256 on startup.*

Types, instances, variables

We differentiate between **type**, **instance**, and **variable**!

```
1 a = list()
2 b = a
3 b.append(1)
4 b = 'this is a string'
```

Lines 1-3: Instance of type `list` is assigned to variables `a` and `b`.

Line 4: Variable `b` refers now to a new string instance

Type conversion

- Python is smart in converting basic data types
- `int(.)`, `float(.)`, `tuple(.)`, …
- Everything evaluates to a Boolean value
  - Boolean conversion is even performed implicitly

Quiz

Which of the following are valid ways to specify strings in Python:

```
"test"      'test'      "foo'bar"      'foo'bar'
```

True or *false*?

- “In a dictionary, values are accessed by their position.”
- “A variable can only reference a single instance at a time.”
- “Data types are placeholders for instances.”
- “Instances are placeholders for data types.”
- “The expression `bool('None')` evaluates to `False`.”

Quiz

Which of the following are valid ways to specify strings in Python:

`"test"` ✔ `'test"` `"foo'bar"` ✔ `'foo'bar'`

True or *false*?

- “In a dictionary, values are accessed by their position.” false
- “A variable can only reference a single instance at a time.” true
- “Data types are placeholders for instances.” false
- “Instances are placeholders for data types.” false
- “The expression `bool('None')` evaluates to `False`.” false

Programming Basics

Data Types & Mutability

Evaluation Order

Conditions & Comparisons

Operator precedence

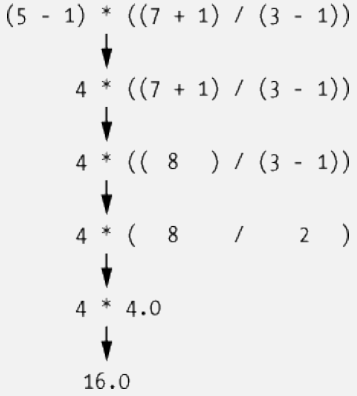
Parentheses	`( ... )`
Exponents	`**`
Multiplication and **D**ivision	`* / // %`
Addition and **S**ubstraction	`+ -`

https://en.wikibooks.org/wiki/Python_Programming/Basic_Math

UNIVERSITÄT BIELEFELD

Expression evaluation

Evaluation: operator precedence + **left-to-right**



Automate the Boring Stuff with Python - Al Sweigart (CC-BY-NC-SA 3.0) chapter 1, figure 1-1, https://automatetheboringstuff.com/chapter1/

Operator Precedence

low

high

Operator	Description
`=, +=, -=, |=, ...`	Assignment expression
`lambda`	Lambda expression
`if - else`	Conditional expression
`or`	Boolean OR
`and`	Boolean AND
`not x`	Boolean NOT
`in, not in, is, is not, <, <=, >, >=, !=, ==`	Comparisons, including membership tests and identity tests
`|`	Bitwise OR
`^`	Bitwise XOR
`&`	Bitwise AND
`<<, >>`	Shifts
`+, -`	Addition and subtraction
`*, @, /, //, %`	Multiplication, matrix multiplication, division, floor division, remainder 5
`+x, -x, ~x`	Positive, negative, bitwise NOT
`**`	Exponentiation 6
`await x`	Await expression
`x[index], x[index:index], x(arguments...), x.attribute`	Subscription, slicing, call, attribute reference
`(expressions...), [expressions...], key: value..., expressions...`	Binding or parenthesized expression, list display, dictionary display, set display

Quiz

- What is the value of the expression `1 + 2 ** 3 * 4`?
- Which of the following operators has the lowest precedence?

  `and` `+` `**` `%` `not`

- Which operation of the expression `'Tiger'[4] + 'oa'* 4 + 'r'` is executed first?

  `'oa' * 4` `'Tiger'[4] + 'oa'` `'Tiger'[4]` `4 + 'r'`

source (in part): `https://realpython.com/quizzes`

Quiz

- What is the value of the expression `1 + 2 ** 3 * 4`? 33
- Which of the following operators has the lowest precedence?

  `and` ✔ `+` `**` `%` `not`

- Which operation of the expression `'Tiger'[4] + 'oa'* 4 + 'r'` is executed first?

  `'oa' * 4` `'Tiger'[4] + 'oa'` `'Tiger'[4]` ✔ `4 + 'r'`

source (in part): `https://realpython.com/quizzes`

Programming Basics

Data Types & Mutability

Evaluation Order

Conditions & Comparisons

Conditional statements: `if/else` clause

```
if «Boolean expression»:
␣␣␣␣«statement»
```

⚠ Mind the indention!

OR

```
if «Boolean expression»:
␣␣␣␣«statement»
else:
␣␣␣␣«alternative statement»
```

Conditional statements: if/else

```
1 a = True
2 if a:
3     print('a is True')
4
5     if 'this is a text':
6         print('another true statement')
```

Conditional statements: `if/else`

```
1 a = True
2 if a:
3     print('a is True')
4 else:
5     print('a is False')
```

Boolean operators and comparisons

Elementary logic: `and`, `or`, `not`

Variables		Boolean expression		
`a`	`b`	`not a`	`a and b`	`a or b`
`False`	`False`	`True`	`False`	`False`
`False`	`True`	`True`	`False`	`True`
`True`	`False`	`False`	`False`	`True`
`True`	`True`	`False`	`True`	`True`

Comparisons: Operators

- `==` “is equal/equivalent to”
- `!=` “is not equal/equivalent to”
- `>` “is larger than”
- `<` “is is smaller than”
- `>=` “is larger or equal to”
- `<=` “is smaller or equal to”
- `is` “is identical instance of”
- `is not` “is not identical instance of”
- `in` “is contained in collection”
- `not in` “is not contained in collection”

Conditional execution based on comparisons

```
1 a = 4.0
2 b = 2.0
3 if not a > b:
4     print('true statement!')
```

Conditional execution based on comparisons

```
1 a = 'this is a text'
2 b = 'this is a text'
3 if a >= b:
4     print('true statement!')
```

Conditional execution based on comparisons

```
1 a = 'this is a text'
2 if a:
3     print('true statement!')
```

Conditional execution based on comparisons

```
1 a = list()
2 b = list()
3 if a is b:
4     print('true statement!')
```

Conditional execution based on comparisons

```
1 a = list()
2 b = list()
3 if a == b:
4     print('true statement!')
```

Conditional execution based on comparisons

```
1 a = list()
2 b = 1
3 if b not in a:
4     print('b is contained in collection a')
5 else:
6     print('b is not contained in collection a')
```

Recap

Summary

- Computer architecture and Python compiler
- Python data types: `int`, `float`, `str`, `tuple`, `list`, `dict`, …
- Operator precedence
- `if`/`else` clause
- Comparators: `==`, `!=`, `>`, `<`, `is`, `in`, …

What comes next?

- Familiarize yourself with Spyder
- Write your first program!
- Due date for this week's exercises is **Saturday, May 2, 2020**.

Next lecture: Programming & Python basics continued …

UNIVERSITÄT BIELEFELD

Spyder