PROGRAM'S ANATOMY

THE LIFI-PROJECT

DISCLAIMER

The following slides are for presentation purposes only. They contain mostly visuals and are not meant to as a script for studying. Please always watch the video or listen to the audio along with these slides and read the respective lessons in the online script.

Please consider the environment before printing these slides.

Always use the link to the original slides to access the latest version. The slides are likely to change during a semester.

TERMS

CAN YOU EXPLAIN?

Python

Programming Language

API

IDE

- ✓ Syntax Highlighting
- ✓ Code Completion
- ✓ Debugging
- ✓ Warnings / Errors
- ✓ Code formatting

GUI

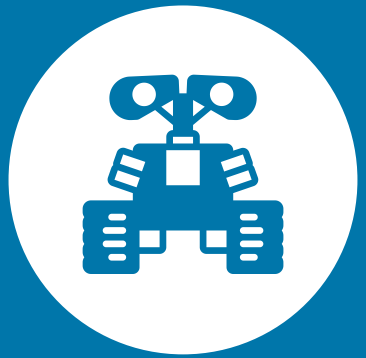
File Extension

File Path

Folder

Operating System (OS)

PROGRAM'S ANATOMY



Can you approximate the square root of a given positive number?

EXPLAIN HOW!

A finite-length rule consisting of individual instructions is called an **algorithm**.

Source: Vornberger, O., Algorithms and Data Structures, Lecture Notes (http://www-lehre.inf.uos.de/~ainf/2013/PDF/skript.pdf)
Translated from German using DeepL

ALGORITHM

PROBLEM-SOLVING

Algorithm = Procedure description for solving a problem.

Program = Algorithm formulated in a programming language.

FROM ALGORITHM TO PROGRAM

EXAMPLE

Algorithm

A recipe to solve a problem

1. Get a number X from user
2. Check if X is positive
3. Set A = X/2 and B = X/A
4. Repeat until |A-B| is less than 0.00001
    4a. Set A = (A+B)/2
    4b. Set B = X/A
5. Give A as the result

Program

Implementation in a programming language

examples > square_root.py > ...

```python
import sys
print("I can calculate square roots!")

number = input("A number, please: ")
number = int(number)

if number < 0:
    print("Cannot extract roots from negative numbers.")
    sys.exit()

a = number / 2
b = number / a

while(abs(a - b) > 0.00001):
    a = (a + b) / 2
    b = number / a

print(f"The square root of { number } is { a }")
```

1. Commands
2. Variables
3. Loops
4. Control Structures
5. Functions

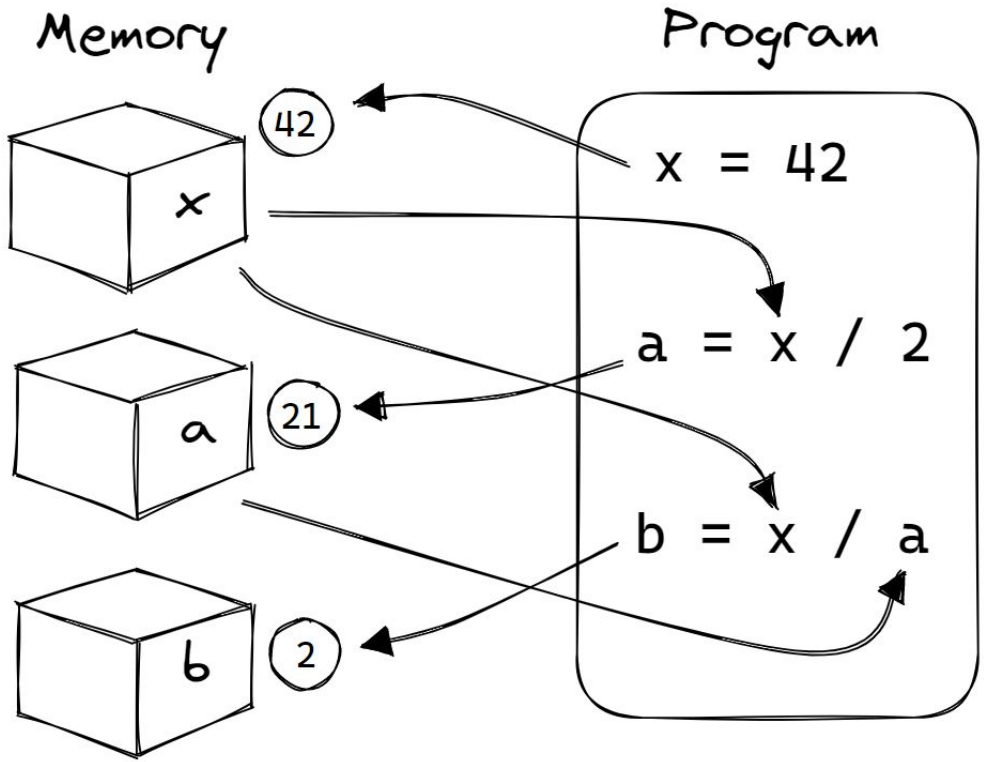
Internal or built-in functions

Functions from external modules

User-defined functions

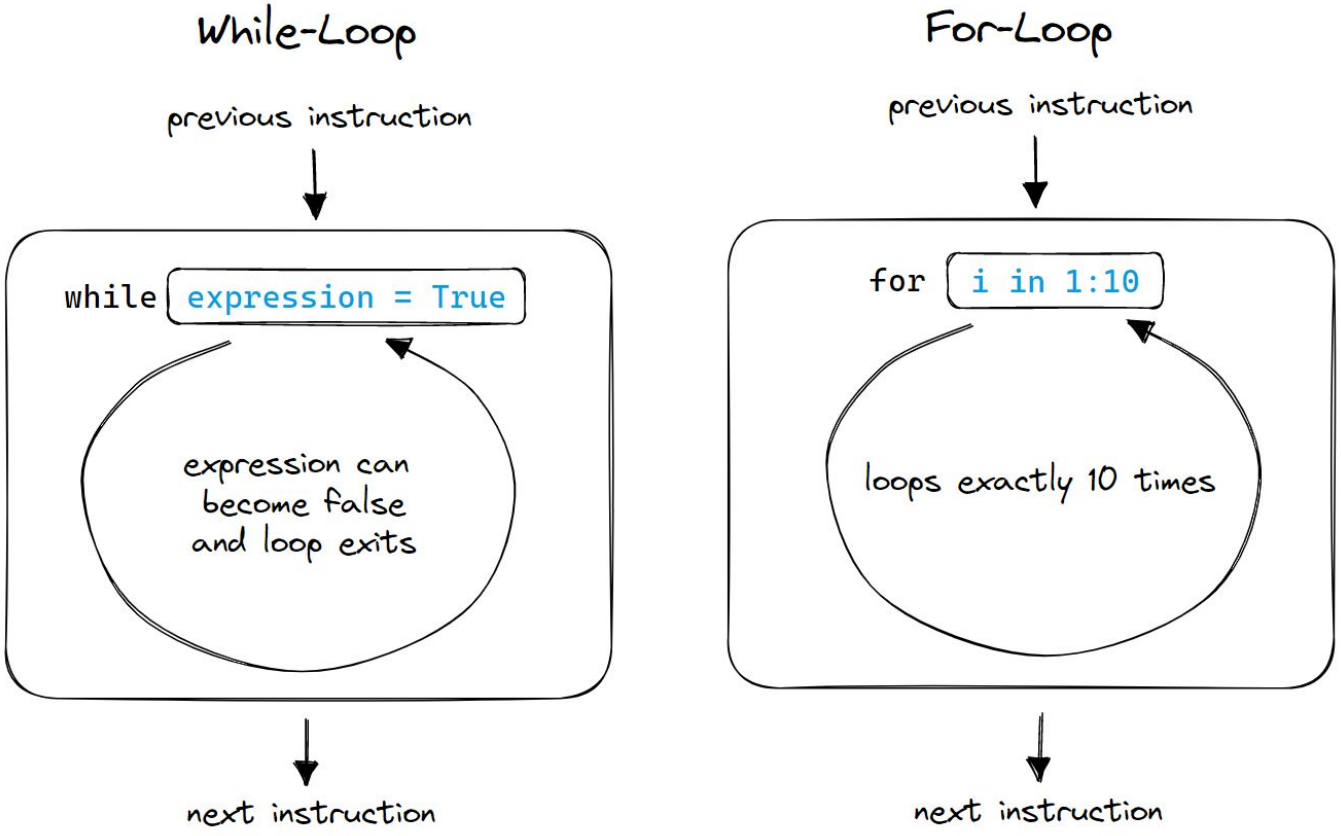
5 TYPES OF INSTRUCTIONS

VARIABLES (2/5)



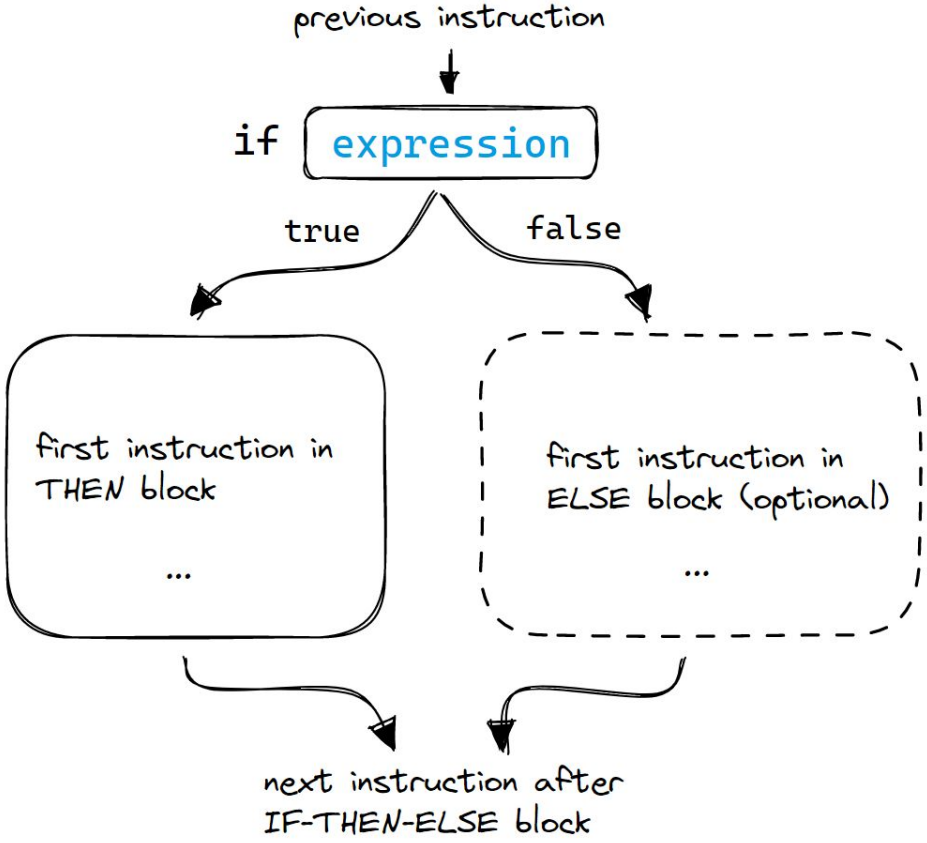
5 TYPES OF INSTRUCTIONS

LOOPS (3/5)



5 TYPES OF INSTRUCTIONS

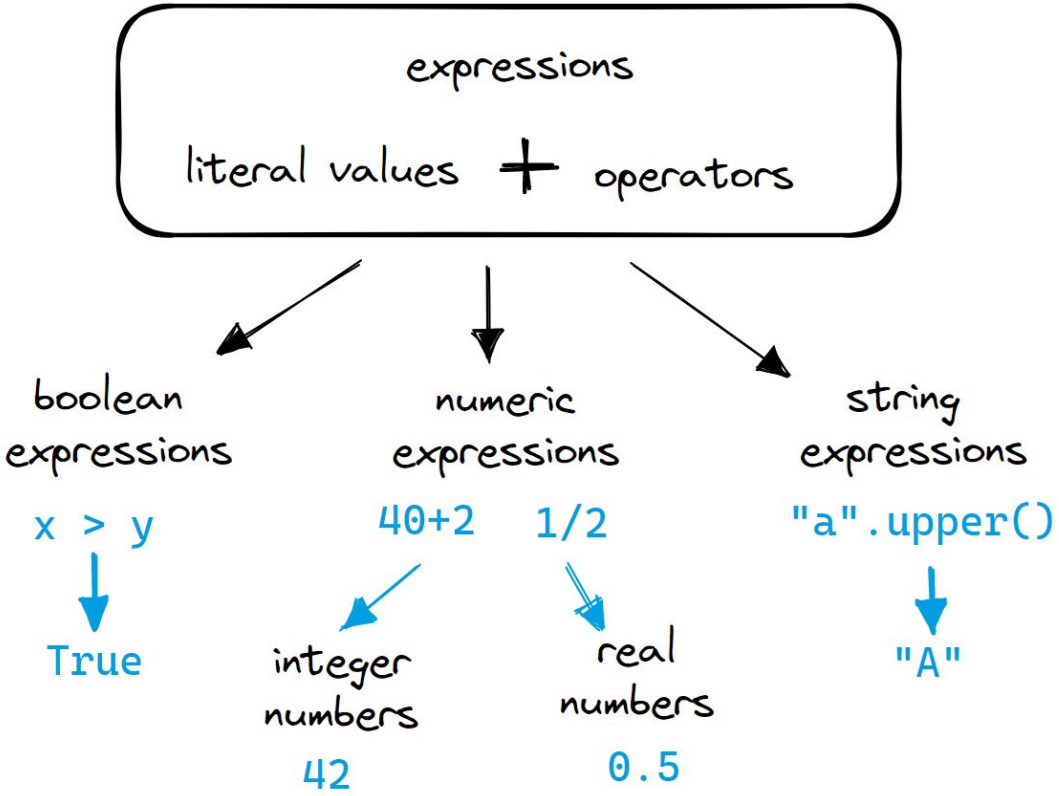
CONTROL STRUCTURES (4/5)



Code block we can reuse by its name

Optional parameters and return value

EXPRESSIONS



Instructions in action to solve a problem