RUSTikales Rust for beginners

Plan for today

Plan for today

1. Introduction

Plan for today

1. Introduction
2. Rust Installation

Plan for today

1. Introduction
2. Rust Installation
3. Development Environment

Plan for today

1. Introduction
2. Rust Installation
3. Development Environment
4. General Info

1. Introduction

- Welcome to this Rust course!

1. Introduction

- Welcome to this Rust course!
- Your life will change forever soon, you will be learning the best language in the world!!!

1. Introduction

- Welcome to this Rust course!
- Your life will change forever soon, you will be learning the best language in the world!!!
  - It's memory safe!
  - It's statically typed!
  - It's fast!
  - Zero Cost Abstraction!

1. Introduction

- Welcome to this Rust course!
- Your life will change forever soon, you will be learning the best language in the world!!!
  - It's memory safe!
  - It's statically typed!
  - It's fast!
  - Zero Cost Abstraction!
- Well, not quite, but you'll see once we get there

1. Introduction

- What will we be learning this semester?

1. Introduction

- What will we be learning this semester?
  - Working with the Rust Compiler rustc

1. Introduction

- What will we be learning this semester?
  - Working with the Rust Compiler rustc
  - Working with the Rust tool cargo
    - cargo init/new
    - cargo build
    - cargo run

1. Introduction

- What will we be learning this semester?
  - Working with the Rust Compiler rustc
  - Working with the Rust tool cargo
  - Working with IDEs
    - Visual Studio Code + rust-analyzer
    - RustRover

1. Introduction

- What will we be learning this semester?
  - Working with the correct tools
  - Basic Rust-Syntax and Rust-specific quirks

1. Introduction

- What will we be learning this semester?
  - Working with the correct tools
  - Basic Rust-Syntax and Rust-specific quirks
    - Basics of Programming
      - Variables
      - Types
      - How do computers execute code?

1. Introduction

- What will we be learning this semester?
  - Working with the correct tools
  - Basic Rust-Syntax and Rust-specific quirks
    - Basics of Programming
    - Control Flow
      - if
      - loop
      - match
      - fn

1. Introduction

- What will we be learning this semester?
  - Working with the correct tools
  - Basic Rust-Syntax and Rust-specific quirks
    - Basics of Programming
    - Control Flow
    - Data Structures
      - struct
      - enum

1. Introduction

- What will we be learning this semester?
  - Working with the correct tools
  - Basic Rust-Syntax and Rust-specific quirks
    - Basics of Programming
    - Control Flow
    - Data Structures
    - What makes Rust different from other languages
      - Ownership
      - Borrow Checking
      - Option<T> and Result<T,E>

1. Introduction

- What will we be learning this semester?
  - Working with the correct tools
  - Basic Rust-Syntax and Rust-specific quirks
  - Dipping into advanced-ish topics
    - Traits
    - Generics
    - Third-party libraries

1. Introduction

- What will we be learning this semester?
  - Working with the correct tools
  - Basic Rust-Syntax and Rust-specific quirks
  - Dipping into advanced-ish topics
- What I will not cover
  - unsafe
  - async
  - Macros
  - Multithreading
  - Functional Programming

1. Introduction

- Where can I find resources?
  - Official Rust website:
    - https://www.rust-lang.org/learn/
  - Rust Book:
    - https://doc.rust-lang.org/book/
  - Rustlings (official Rust exercises):
    - https://rustlings.cool/
  - Repo for this course:
    - https://github.com/pfhaupt/progkurs/
    - subdirectory rust-beginner

2. Rust Installation

2. Rust Installation

Linux/macOS

- Go to `rust-lang.org`
- Click `Get Started`
- Run this command in your terminal

It looks like you're running macOS, Linux, or another Unix-like OS. To download Rustup and install Rust, run the following in your terminal, then follow the on-screen instructions. See "Other Installation Methods" if you are on Windows.

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

- You may need to also install a linker
  - you'll see later when the example fails
- You're ready to go!

Windows

- Go to `rust-lang.org`
- Click `Get Started`
- Download the Installer

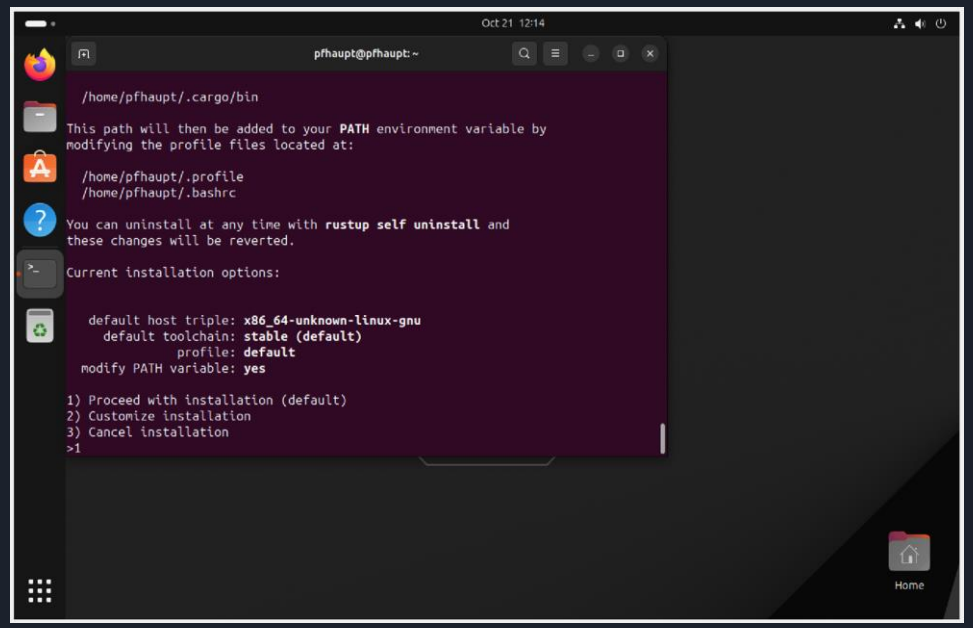
It looks like you're running Windows. To start using Rust, download the installer, then run the program and follow the onscreen instructions. You may need to install the Visual Studio C++ Build tools when prompted to do so. If you are not on Windows see "Other Installation Methods".

DOWNLOAD RUSTUP-INIT.EXE (32-BIT) | DOWNLOAD RUSTUP-INIT.EXE (64-BIT)

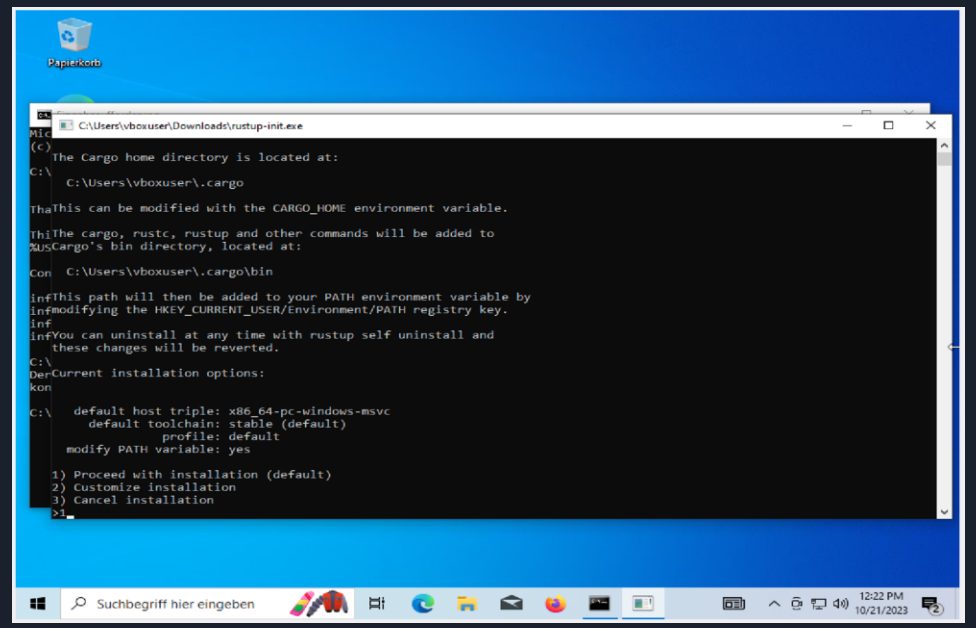
- Run the Installer
- You may need to also install MSVC Tools
  - Installer will tell you, Quick Install is okay
  - Pray that eduroam doesn't blacklist the MSVC Installer again :^)

2. Rust Installation

Linux/macOS



Windows



- If you see this, you're almost done! It will now install all necessary tools.
- Restart your terminal, and you'll be able to use rustc and cargo!

2. Rust Installation

– if Installation was successful, you should be able to run the following commands:

```
C:\>rustup --version
rustup 1.27.0 (bbb9276d2 2024-03-08)
info: This is the version for the rustup toolchain manager, not the rustc compiler.
info: The currently active `rustc` version is `rustc 1.76.0 (07dca489a 2024-02-04)`

C:\>cargo --version
cargo 1.76.0 (c84b36747 2024-01-18)

C:\>rustc --version
rustc 1.76.0 (07dca489a 2024-02-04)
```

2. Rust Installation

- if any of those commands failed, we must troubleshoot now

2. Rust Installation

- if any of those commands failed, we must troubleshoot now
- common errors:
  - `Linker cc not found`
    - Linux → sudo apt install gcc
    - macOS → brew install gcc
    - Windows → shouldn't happen, the Installer installed the MSVC toolchain :^)
  - eduroam may block downloading the Installer → you'd need to try again at home

2. Rust Installation

- To test if everything is set up properly, run those commands:
  - Create a directory of your choice
    - Either via file manager, or terminal
  - Open that directory in a terminal
  - type in cargo new test_program
  - Navigate into that directory with cd test_program
  - type in cargo run
  - if you see Hello, world!, you're ready to go!

3. Development Environment

- Technically it doesn’t matter, you could even use Ed, Notepad, Word…

3. Development Environment

- Technically it doesn’t matter, you could even use Ed, Notepad, Word…
- There are some IDEs and Editors that make programming [in Rust] easier

3. Development Environment

- Technically it doesn’t matter, you could even use Ed, Notepad, Word…
- There are some IDEs and Editors that make programming [in Rust] easier
  - JetBrains RustRover → Free license for CS-students with a University-email
  - Emacs, Vim → For those who prefer terminal editors
  - Visual Studio Code → What I’ll be using

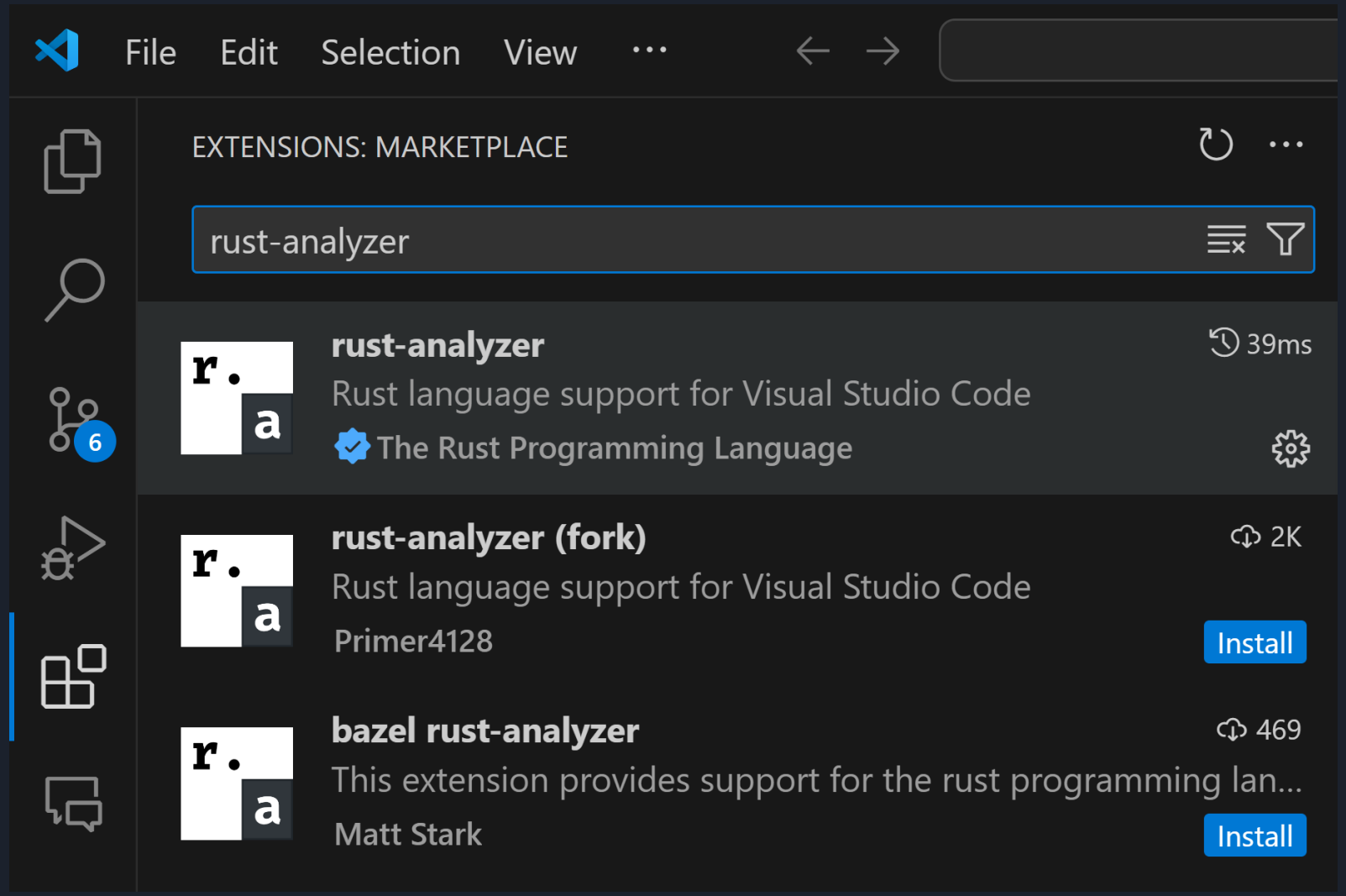
3. Development Environment

- Technically it doesn't matter, you could even use Ed, Notepad, Word…
- There are some IDEs and Editors that make programming [in Rust] easier
- VSC → https://code.visualstudio.com/
  - Windows → Simply follow the Installer
  - Linux → https://code.visualstudio.com/docs/setup/linux
  - macOS → https://code.visualstudio.com/docs/setup/mac

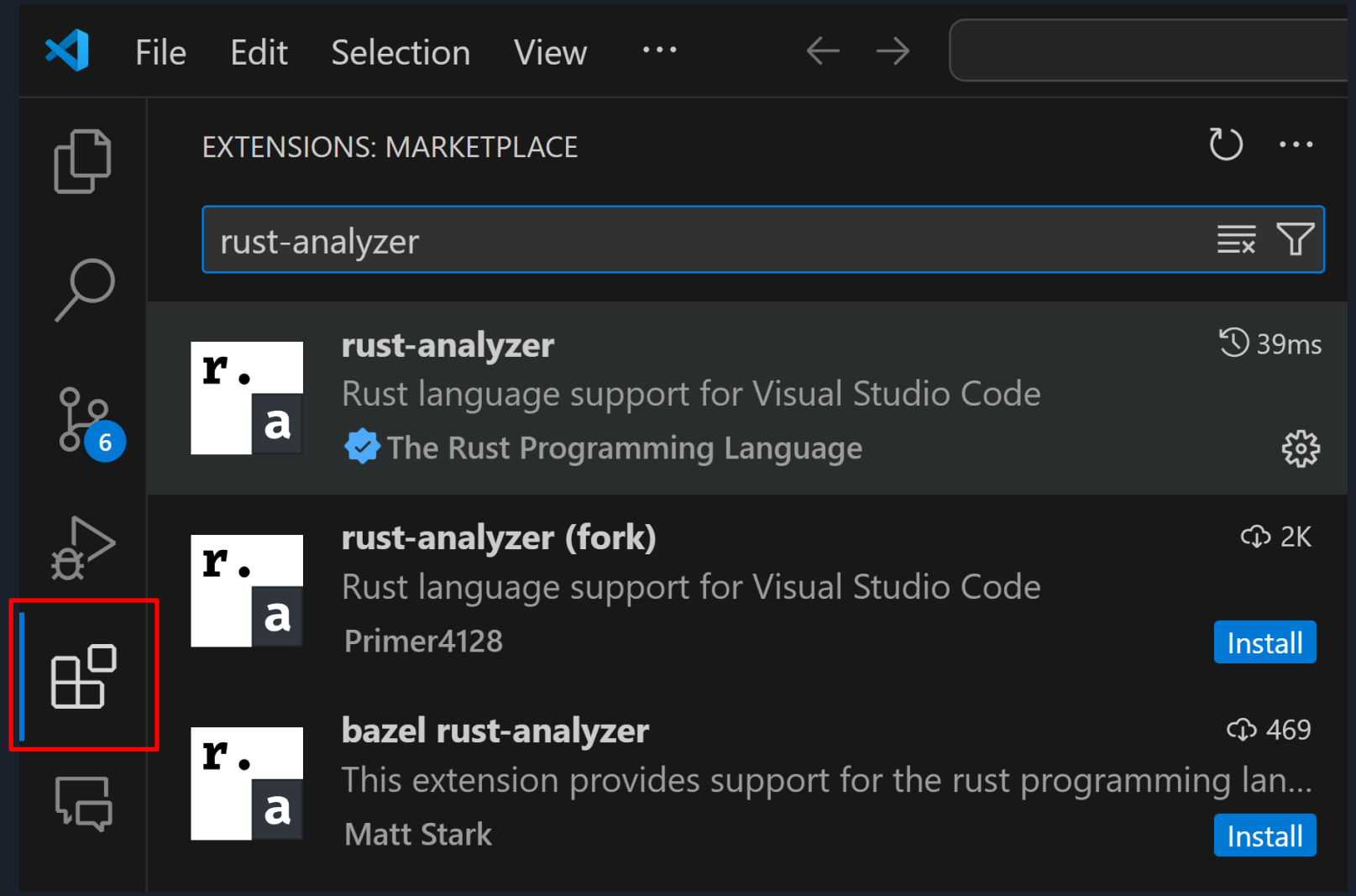
3. Development Environment

- Visual Studio Code itself does not have builtin Rust-Support, only Syntax Highlight – It's just a Text Editor
  - Using extensions, we can fix that

3. Development Environment

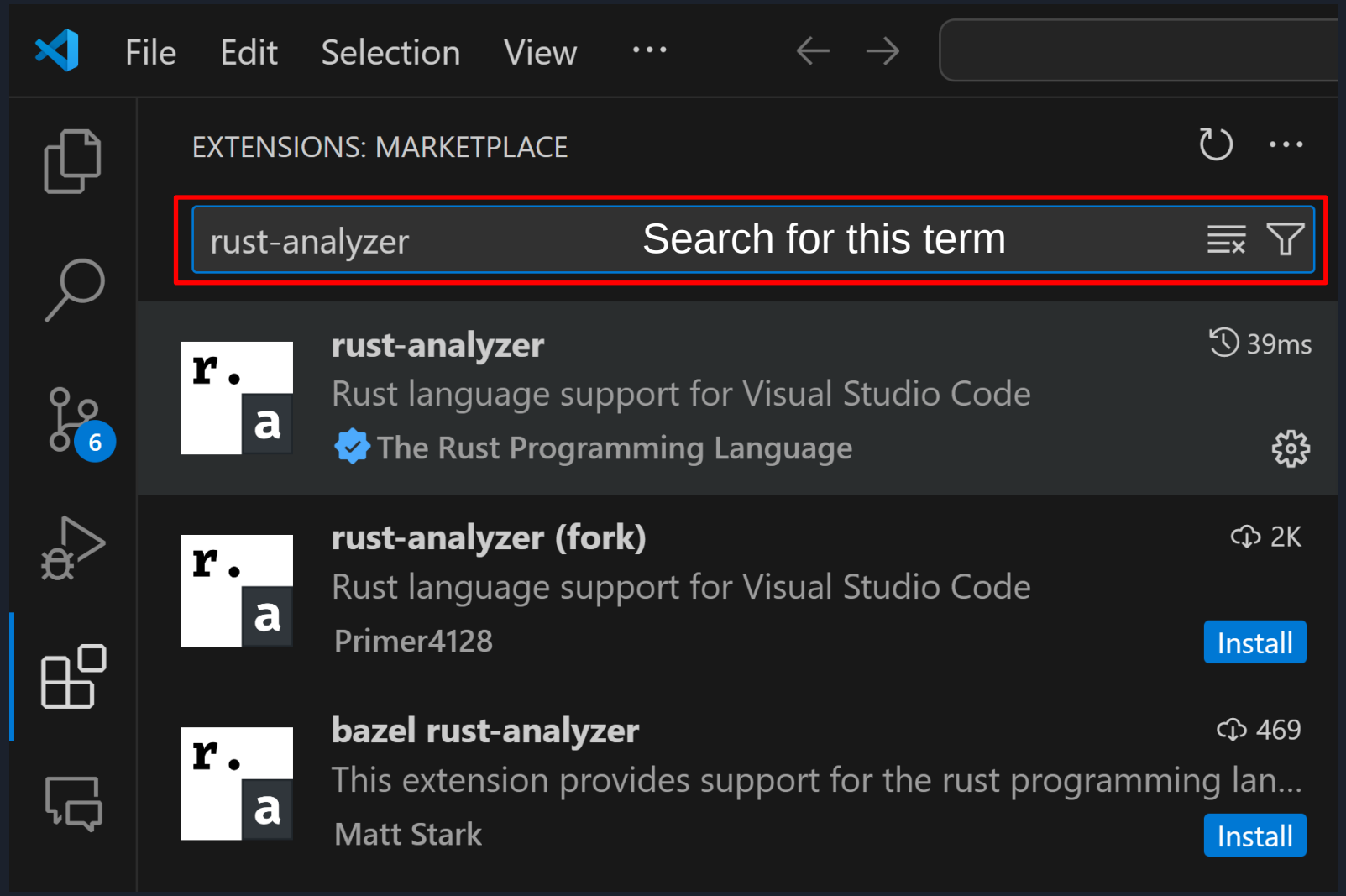


3. Development Environment

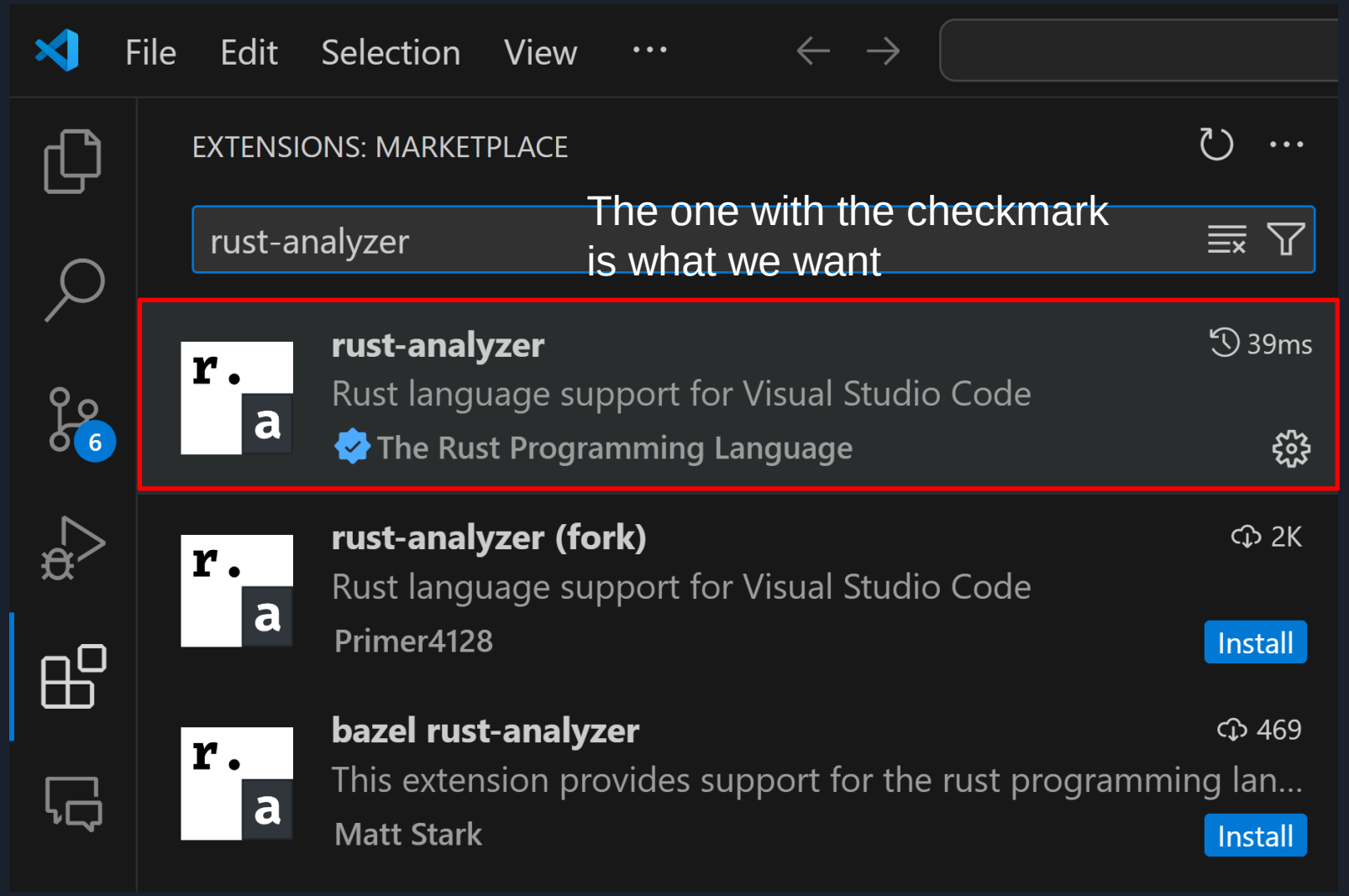


Extensions

3. Development Environment

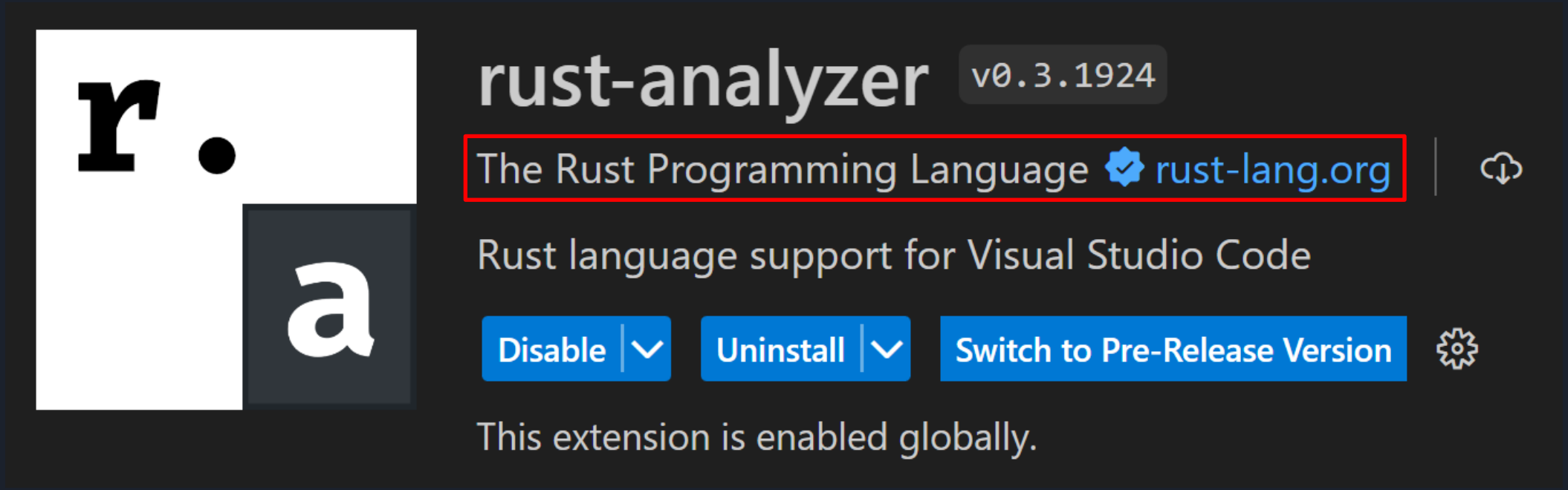


3. Development Environment

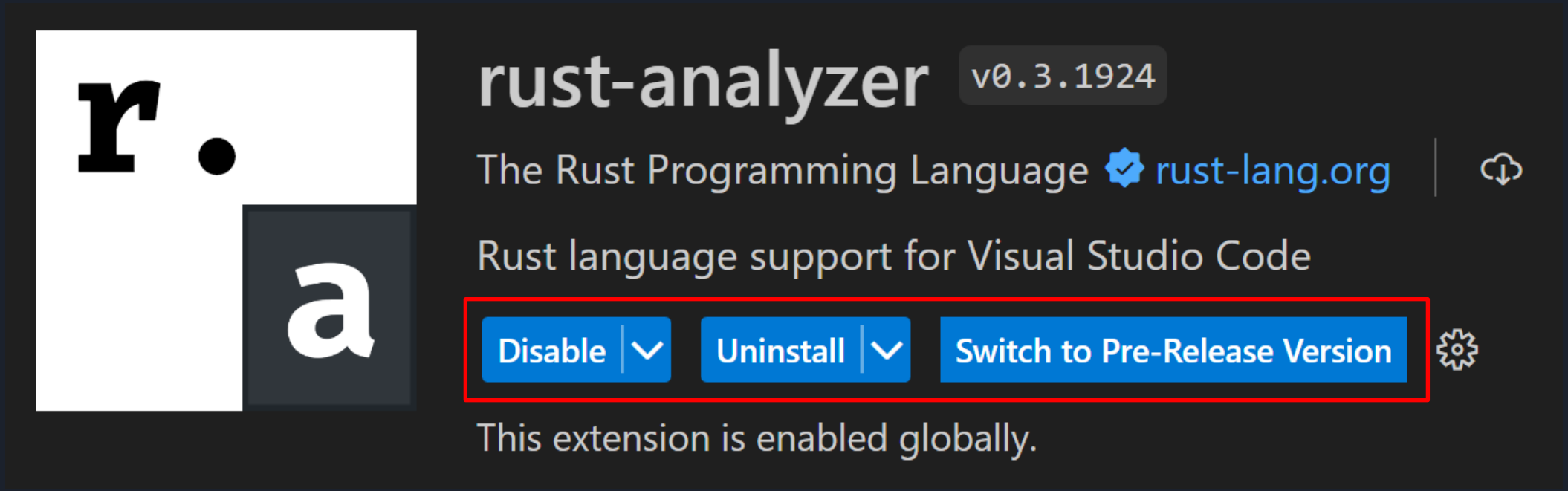


3. Development Environment

As extensions can run arbitrary code, make sure it‘s the right one :^)



3. Development Environment



Somewhere here should be an Install-button
Restart VSC if necessary

3. Development Environment

```
▶ Run | Debug
fn main() {
    let a: i32 = 10;
    let b: &mut i32 = &mut a;
}
```

```
cannot mutate immuta
analyzer(E0384)

cannot borrow `a` as
declared as mutable
cannot borrow as mut
compiler diagnostic)

main.rs(17, 9): cons
mutable: `mut `
```

3. Development Environment

We can run our code from within VSC

```
▶ Run | Debug
fn main() {
    let a: i32 = 10;
    let b: &mut i32 = &mut a;
}
```

```
cannot mutate immuta
analyzer(E0384)

cannot borrow `a` as
declared as mutable
cannot borrow as mut
compiler diagnostic)

main.rs(17, 9): cons
mutable: `mut `
```

3. Development Environment

```
▶ Run | Debug
fn main() {
    let a: i32 = 10;
    let b: &mut i32 = &mut a;
}
```

```
cannot mutate immuta
analyzer(E0384)

cannot borrow `a` as
declared as mutable
cannot borrow as mut
compiler diagnostic)

main.rs(17, 9): cons
mutable: `mut `
```

Direct error reporting in our editor, without compiling ourselves

3. Development Environment

Diagnostics when hovering over the red swiggly lines of death :^)

```
cannot mutate immuta
analyzer(E0384)

cannot borrow `a` as
declared as mutable
cannot borrow as mut
compiler diagnostic)

main.rs(17, 9): cons
mutable: `mut `
```

```
▶ Run | Debug
fn main() {
    let a: i32 = 10;
    let b: &mut i32 = &mut a;
}
```

4. General Info

- Slides are available at the mentioned Github repository

4. General Info

- Slides are available at the mentioned Github repository
- I highly recommend using the default git way of getting the repository
  - git is very important, it's best to start early
  - Because I am still changing slides, regular updates are recommended
  - Initial Step:
    - Go to the Github repository
    - Click the green Code button, copy the HTTPS-url to your clipboard
    - Go to a directory of your choice, open the terminal there
    - type in git clone <paste URL here>
  - In the future, to get the updated slides:
    - Simply go to that directory (or rather, the progkurs directory inside)
    - type in git pull
    - You now have access to the current state of all slides and exercises

4. General Info

- Slides are available at the mentioned Github repository
- Every session will be split into three parts
  - Recap of last session
  - New topics
  - Exercises at the end

4. General Info

- Slides are available at the mentioned Github repository
- Every session will be split into three parts
- Exercises and some Code snippets in future slides will be color coded
  - Green → 0/3 → We have covered the topic already, should be easy enough
  - Yellow → 1/3 → We have just covered the topic, may be hard
  - Red → 2/3 → Same as Yellow, but trickier
  - Purple → 3/3 → We have not covered the topic, but challenges are always fun

4. General Info

- Slides are available at the mentioned Github repository
- Every session will be split into three parts
- Exercises and some Code snippets in future slides will be color coded
- Every session contains a file with exercises called exercises.md
  - Exercises will not be fully compared every time
  - Important points will be mentioned in each Recap
  - Example solutions are in a file called solutions.md

4. General Info

- Slides are available at the mentioned Github repository
- Every session will be split into three parts
- Exercises and some Code snippets in future slides will be color coded
- Every session contains a file with exercises called exercises.md
  - Example solutions are in a file called solutions.md
- Participation and Feedback is very important
  - Basic program stands, but my goal is to teach you Rust the best I can
  - Don’t understand something? Am I too fast? Did I make any mistakes?
    - Just raise your hand, and we can discuss a topic for a while! ☺

5. Next time

- rustc vs cargo
- Basic Types
- Variables
- let vs let mut