

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with faint, lighter blue diagonal stripes.

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"](#).

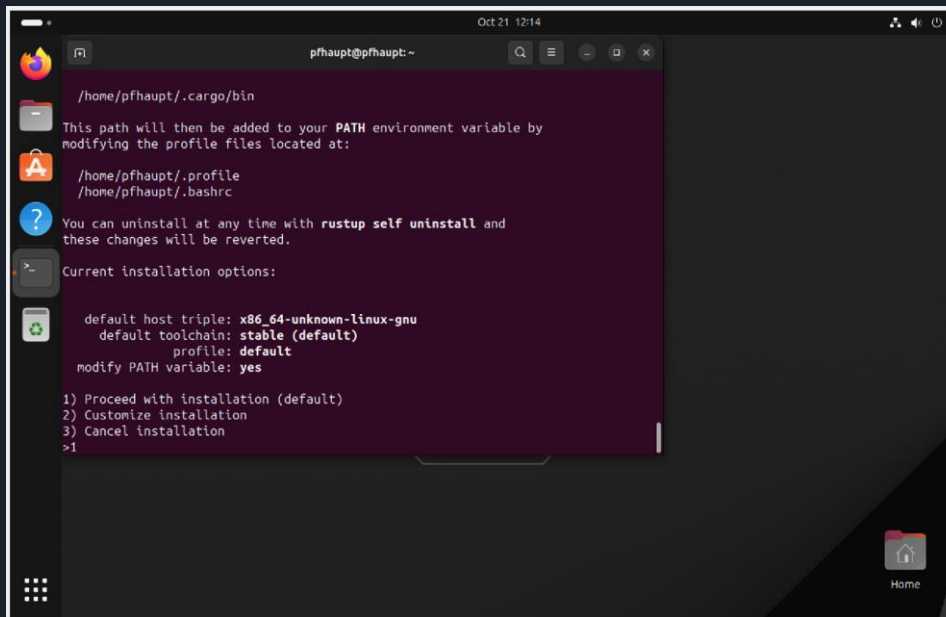
DOWNLOAD RUSTUP-INIT.EXE (32-BIT)

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

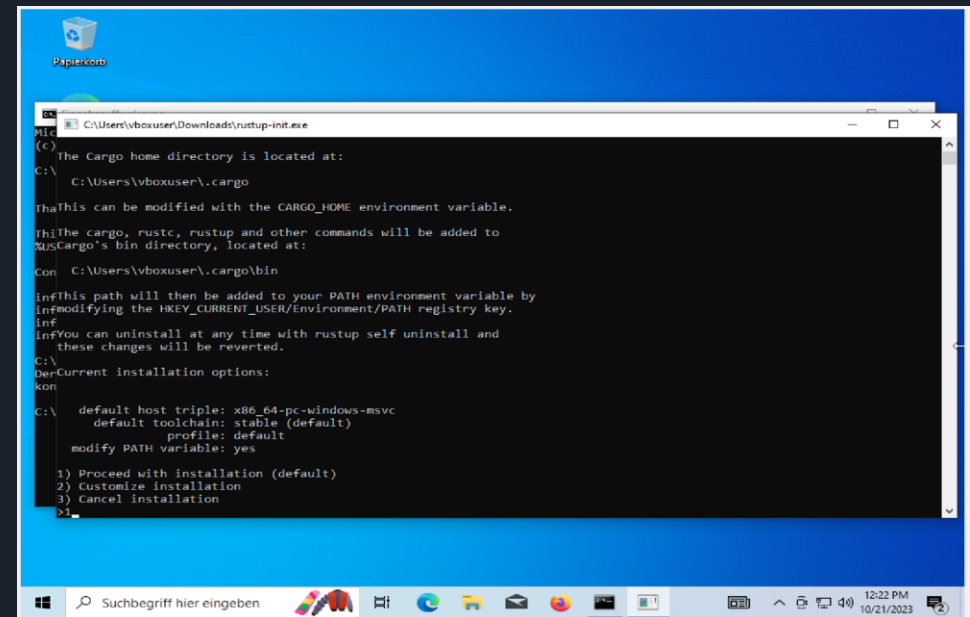
Linux/macOS



A terminal window on a Linux/macOS system showing the rustup installation process. The prompt is 'pfhaupt@pfhaupt:~'. The output shows the cargo bin directory, instructions to add it to the PATH, the profile files being modified, and the current installation options. The user has selected '1) Proceed with installation (default)'.

```
pfhaupt@pfhaupt:~  
/home/pfhaupt/.cargo/bin  
This path will then be added to your PATH environment variable by  
modifying the profile files located at:  
/home/pfhaupt/.profile  
/home/pfhaupt/.bashrc  
You can uninstall at any time with rustup self uninstall and  
these changes will be reverted.  
Current installation options:  
  
default host triple: x86_64-unknown-linux-gnu  
default toolchain: stable (default)  
profile: default  
modify PATH variable: yes  
  
1) Proceed with installation (default)  
2) Customize installation  
3) Cancel installation  
>1
```

Windows



A Windows command prompt window showing the rustup installation process. The prompt is 'C:\Users\vboxuser\Downloads>rustup-init.exe'. The output shows the cargo home directory, instructions to add it to the PATH, the registry key being modified, and the current installation options. The user has selected '1) Proceed with installation (default)'.

```
C:\Users\vboxuser\Downloads>rustup-init.exe  
(c) The Cargo home directory is located at:  
C:\Users\vboxuser\.cargo  
This can be modified with the CARGO_HOME environment variable.  
The cargo, rustc, rustup and other commands will be added to  
Cargo's bin directory, located at:  
C:\Users\vboxuser\.cargo\bin  
This path will then be added to your PATH environment variable by  
modifying the HKEY_CURRENT_USER/Environment/PATH registry key.  
You can uninstall at any time with rustup self uninstall and  
these changes will be reverted.  
Current installation options:  
  
default host triple: x86_64-pc-windows-msvc  
default toolchain: stable (default)  
profile: default  
modify PATH variable: yes  
  
1) Proceed with installation (default)  
2) Customize installation  
3) Cancel installation  
>1
```

- 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

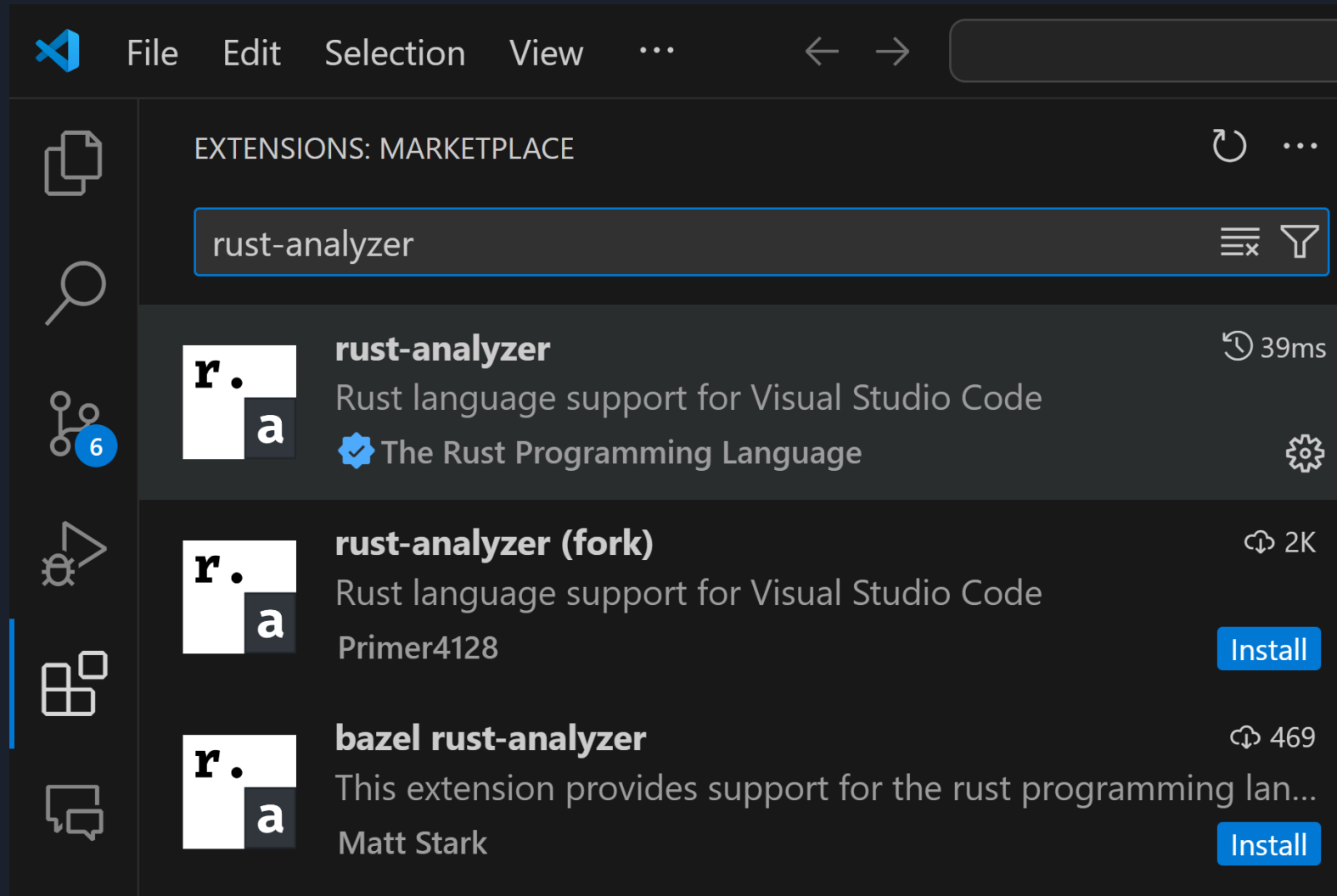
- Technically it doesn't matter, you could even use Ed, Notepad, Word...
- There are some IDEs and Editors that make programming [in Rust] easier
- VSC → <https://code.visualstudio.com/>
 - Windows → Simply follow the Installer
 - Linux → <https://code.visualstudio.com/docs/setup/linux>
 - macOS → <https://code.visualstudio.com/docs/setup/mac>



3. Development Environment

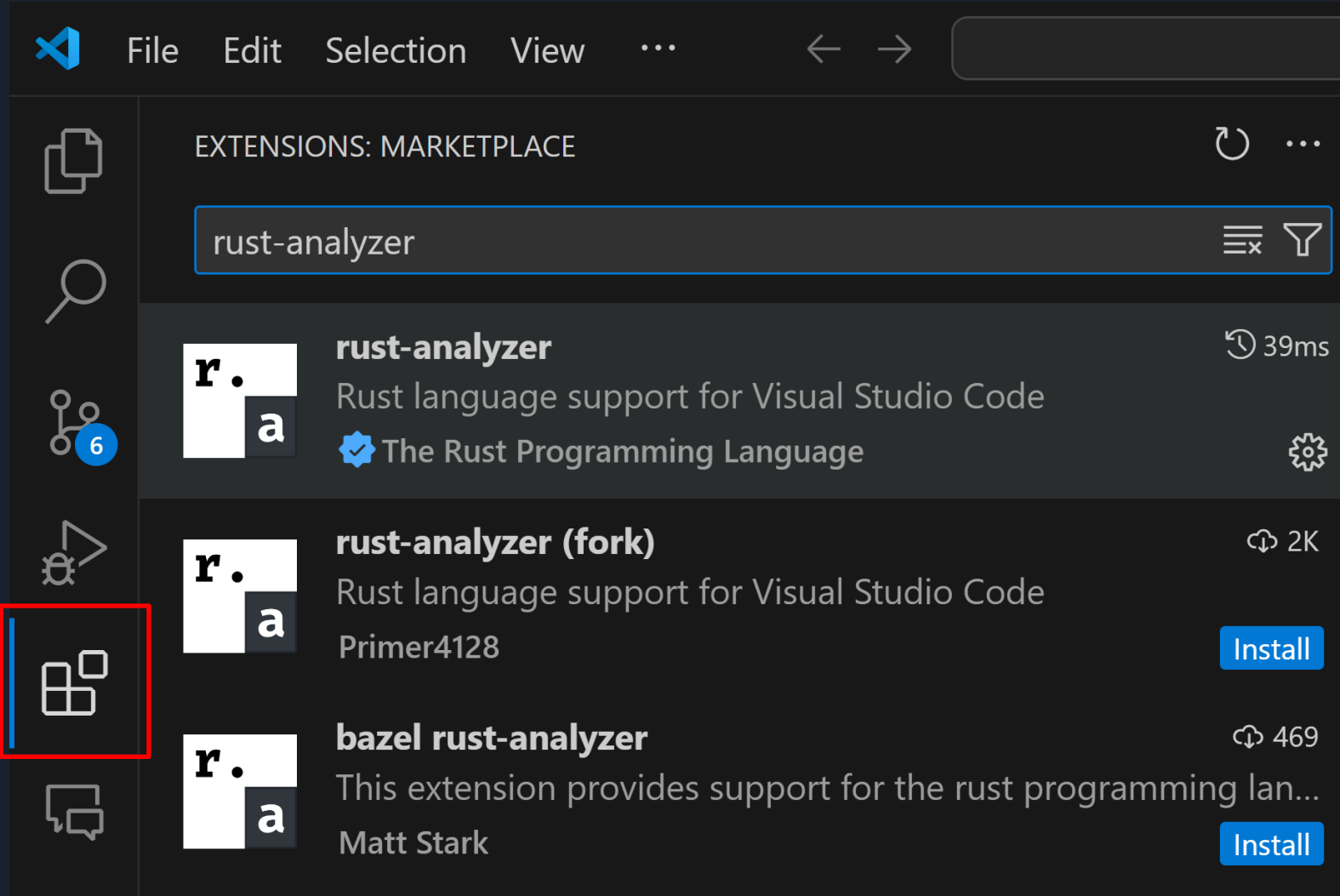
- Visual Studio Code itself does not have builtin Rust-Support, only Syntax Highlight – It's just a Text Editor
 - Using extensions, we can fix that

3. Development Environment



3. Development Environment

Extensions

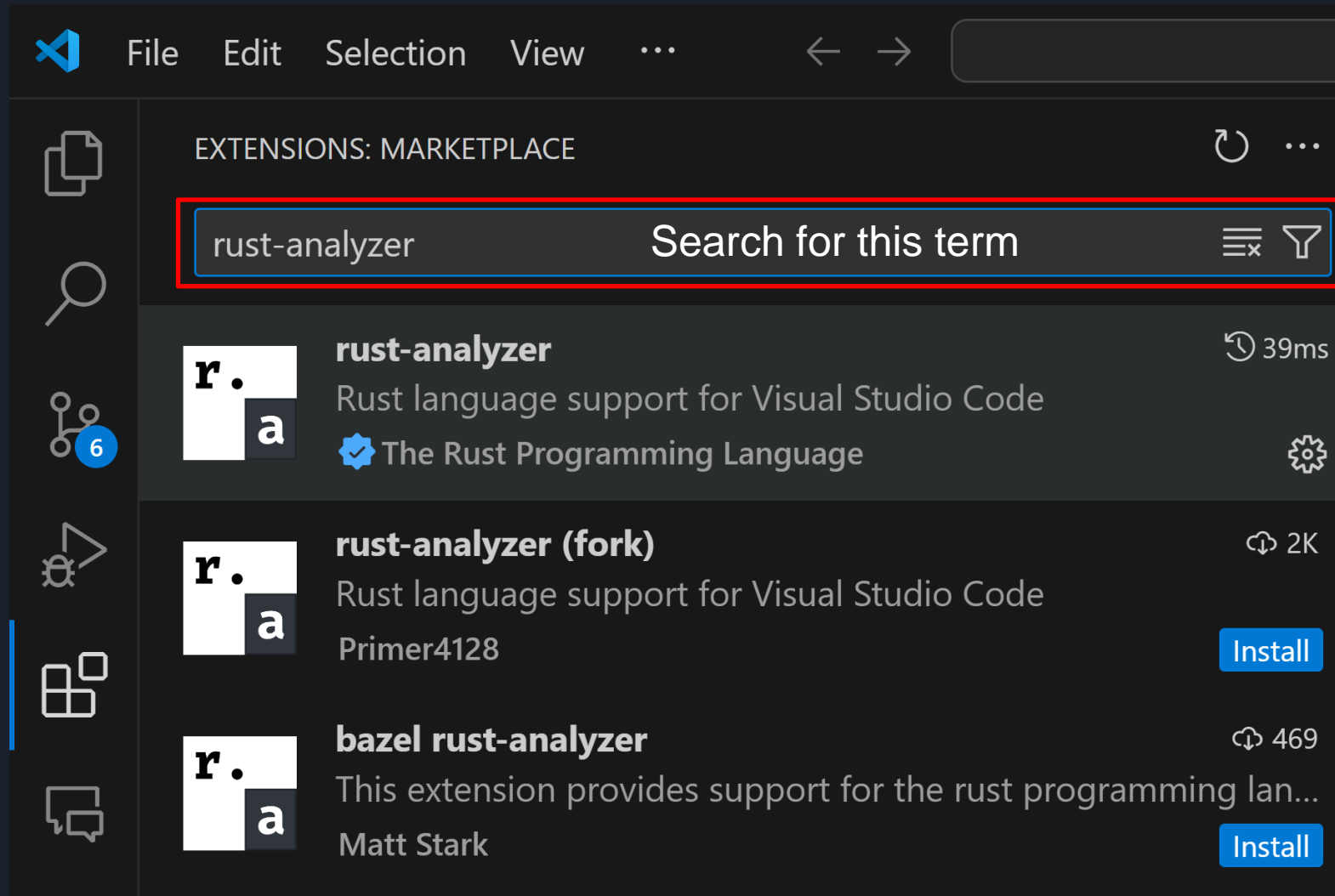


The screenshot shows the Visual Studio Code interface with the 'EXTENSIONS: MARKETPLACE' view. The search bar contains 'rust-analyzer'. Three extensions are listed:

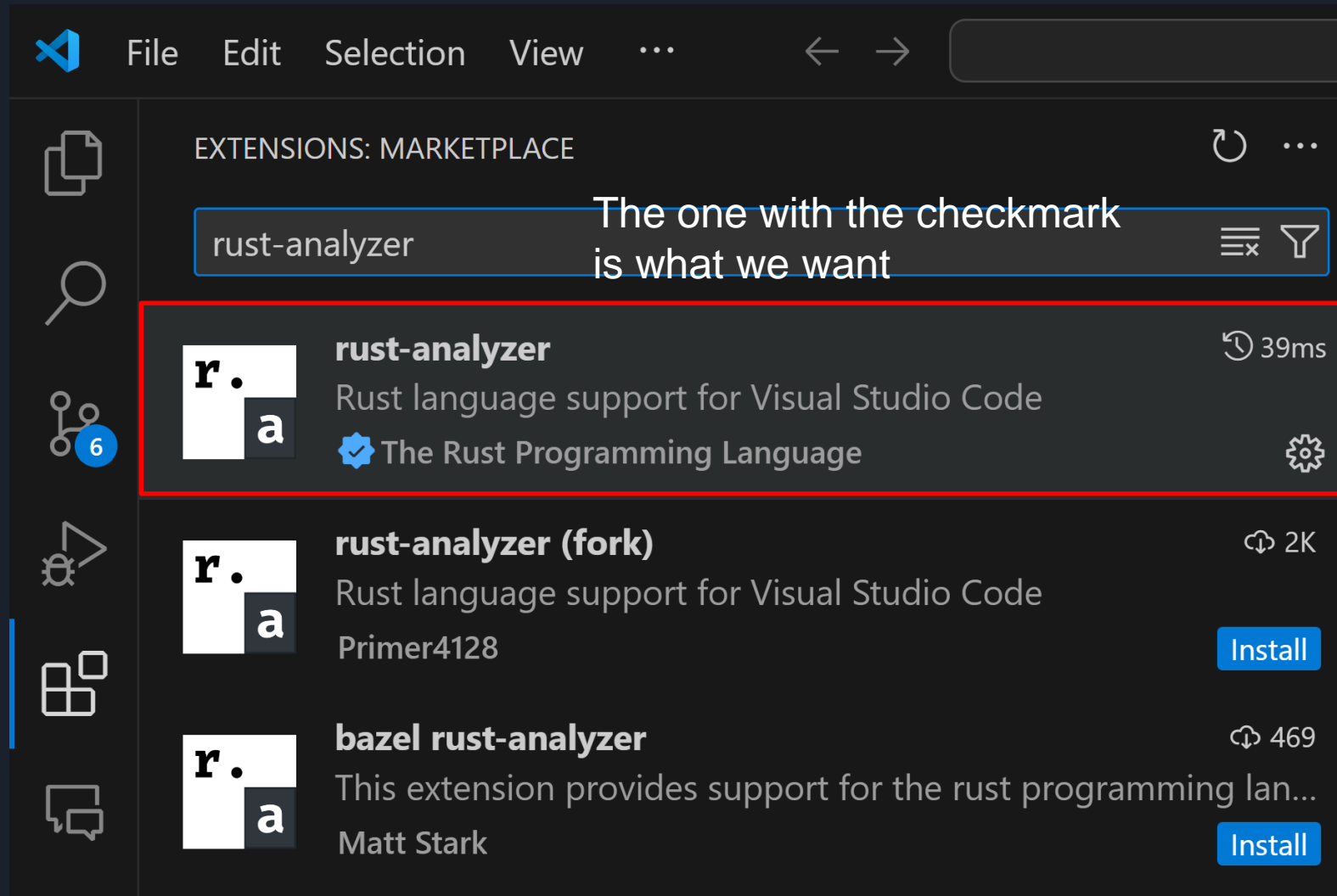
- rust-analyzer** by The Rust Programming Language (39ms). It has a blue checkmark icon and a gear icon.
- rust-analyzer (fork)** by Primer4128 (2K). It has an 'Install' button.
- bazel rust-analyzer** by Matt Stark (469). It has an 'Install' button.

The 'Extensions' sidebar icon is highlighted with a red box.

3. Development Environment

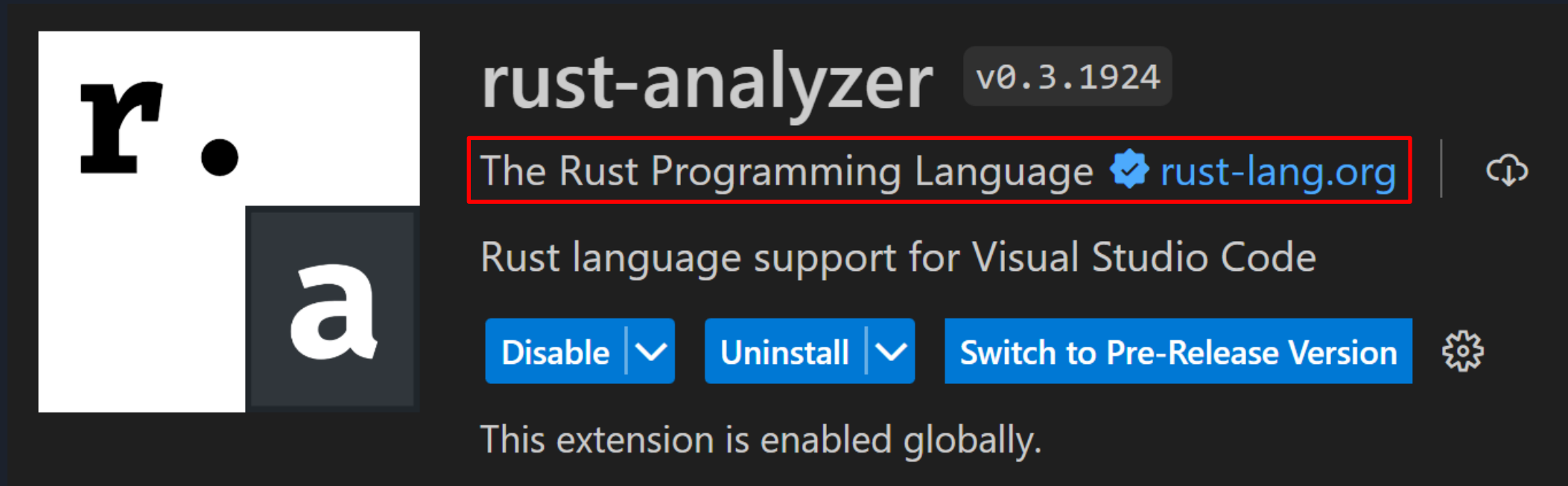


3. Development Environment





3. Development Environment

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






The screenshot shows the 'rust-analyzer' extension page in Visual Studio Code. On the left is the extension's icon, which consists of a white square with a black 'r' and a black dot, and a dark gray square with a white 'a'. To the right of the icon, the extension name 'rust-analyzer' is displayed in a large, bold, white font, followed by the version number 'v0.3.1924' in a smaller font. Below the name, the description 'The Rust Programming Language' is shown in white, followed by a blue checkmark icon and the website 'rust-lang.org' in blue. A red rectangular box highlights this description and website link. To the right of the description is a cloud icon. Below the description, the text 'Rust language support for Visual Studio Code' is displayed in white. Underneath this text are three blue buttons: 'Disable' with a white dropdown arrow, 'Uninstall' with a white dropdown arrow, and 'Switch to Pre-Release Version'. To the right of these buttons is a white gear icon. At the bottom of the extension card, the text 'This extension is enabled globally.' is displayed in white.

rust-analyzer v0.3.1924

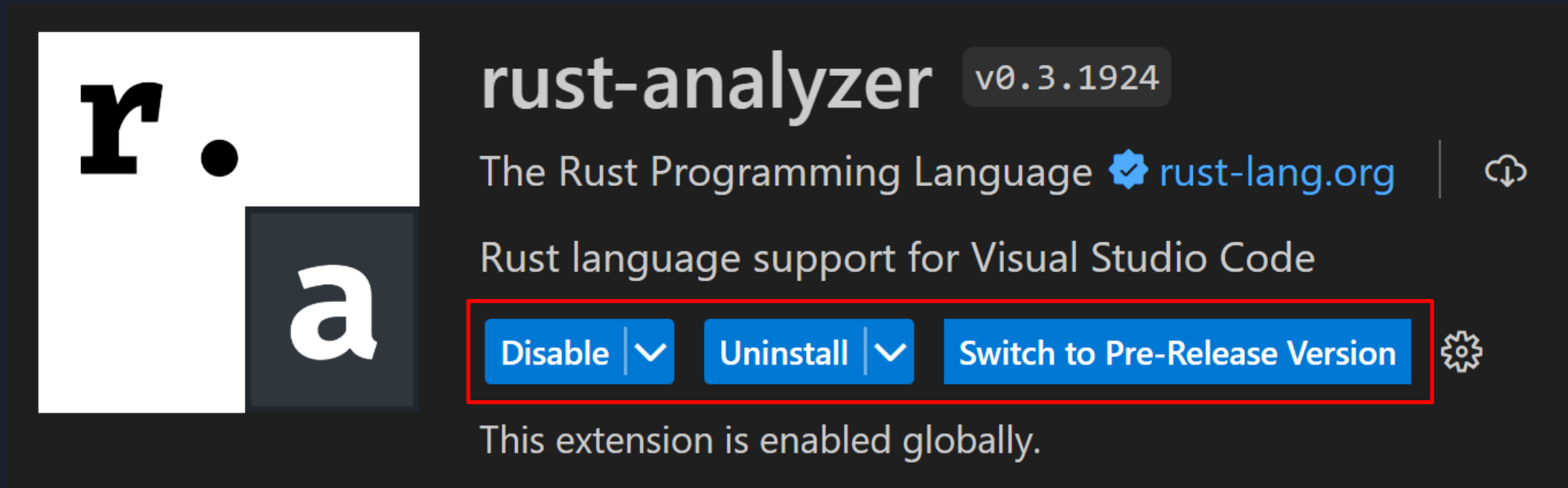
The Rust Programming Language  rust-lang.org | 

Rust language support for Visual Studio Code

[Disable](#) |  [Uninstall](#) |  [Switch to Pre-Release Version](#) 

This extension is enabled globally.

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 immutably
analyzer([E0384](#))

cannot borrow `a` as
declared as mutable
cannot borrow as mutable
[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 immut
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 immutably
analyzer([E0384](#))

cannot borrow `a` as
declared as mutable
cannot borrow as mutable
[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 :^)

► Run | Debug

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

cannot mutate immutable
analyzer([E0384](#))

cannot borrow `a` as
declared as mutable
cannot borrow as mutable
[compiler diagnostic](#))

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



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 **progekurs** 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`