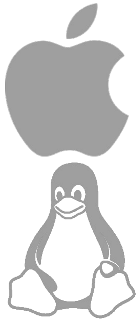
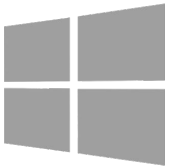
Installing Rust

https://www.rust-lang.org/tools/install



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



See "Other Installation Methods" if you are on Windows.

- On Windows, download and run `rustup-init.exe`.

```
cargo new hello_world
cd hello_world
```

```
.
├── Cargo.toml
└── src
    └── main.rs
```

```
fn main() {
    println!("Hello, world!");
}
```

```
cargo run
```

FizzBuzz

```
for i in 1..11 {
    //Your code here
}
```

Loop through the numbers 1-20

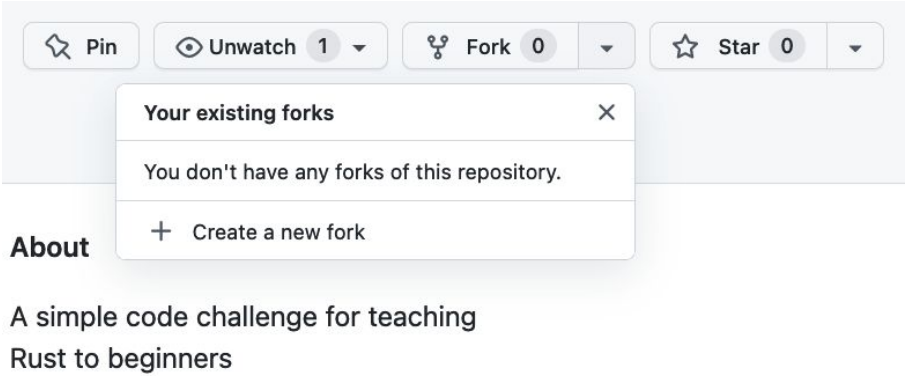
- If the number is divisible by 3, print “Fizz”
- If the number is divisible by 5, print “Buzz”
- If it is divisible by both, print “FizzBuzz”
- If none of the above is true, print nothing.

```
if n < 0 {
    println!("{} is negative", n);
} else if n > 0 {
    println!("{} is positive", n);
} else {
    println!("{} is zero", n);
}
```

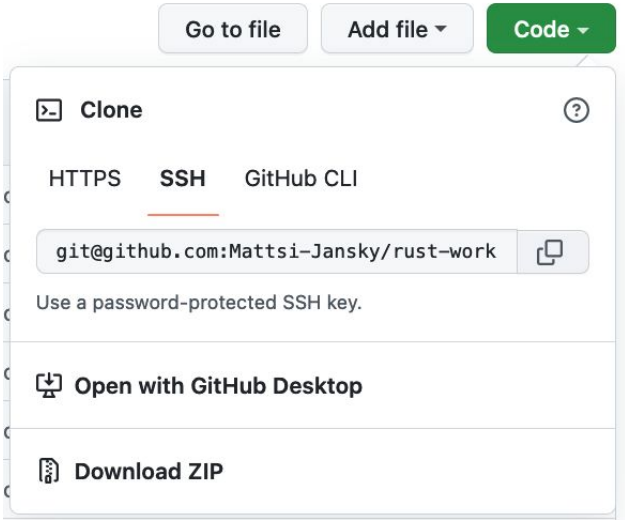
```
let add = 1 + 1; //2
let sub = 6 - 2; //4
let div = 10 / 2; //5
let multi = 5 * 2; //10
let rem = 10 % 8; //2
```

https://github.com/Mattsi-Jansky/rust-workshop

Have a GitHub account? Fork the repo, and clone your fork.



Otherwise, clone the repo with git or download a zip.



Immutability by default

```
let a = 1;
a = 2;
//^ Compile error!

let mut a = 1;
a = 2;
//^ Not a compile error!
```

Return value from function by omitting semicolon

```
fn give_me_a_string() -> String {
  String::new()
}
```

Remember that the grid is one-dimensional, so the elements represent the following:

0	1	2
3	4	5
6	7	8

Step 1 - Rendering

Matching an enum

```
enum Fruit {
  Apple,
  Orange
}

let my_fruit = Fruit::Apple;

match my_fruit {
  Fruit::Apple => { println!("apple") },
  Fruit::Orange => println!("Orange")
}
```

Initialise a struct

```
struct Car {
  passenger: String
}

let my_car = Car {
  passenger: String::from("Mattsi")
};
```

Types

```
(u32) 0; //32-bit unsigned integer
//^ Cannot go below zero
(i32) -1; //32-bit signed integer
//^ Can go below zero
(usize) 0; //Unsigned size integer
//^ Used for collection indexes
//eg myCollection[usize]
(bool) true; //Boolean
(&str) "foobar"; //Immutable string slice
(String) String::from("foobar") //Mutable str
```

Enum tuples

```
enum Error {
  OutOfMemory(u32),
  IoError(String)
}

let io = IoError("not found");
let memory = OutOfMemory(256);
```

Step 1 - Rendering

Iteration

```
for i in iterator {
  //do something with i
}

for n in 1..101 {
  //Do something with n
  //Iterates 1-100
}

loop {
  //Do something until..
  break;
}

while n < 10 {
  //do something
  n++;
}
```

Access struct variable

```
struct Car {
  passengers: Vec<String>
}

impl Car {
  fn remove_passengers(&mut self) {
    self.passengers.pop();
  }
}
```

Adding to a string

```
let mut my_string = String::new();

my_string.push_str("hello world");
//^ String slice
my_string.push('!');
//^ char
```

Step 2 - Validate Moves

Test if variable is enum variant

```
let cell = Cell::Nort;

if matches!(cell, Cell::None) {
  //Do something
}
```