Functions, Comments

main function - entry point of many programs

Declared using the *fn* keyword.

Conventional style - snake case

Example:

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

    another_function();
}

fn another_function() {
    println!("Another function.");
}
```

In the above example another_function is defined after main, we can define before main also. It just needs to be somewhere in the scope of that code.

Parameters/Arguments

must declare the type of each parameter

```
fn another_function(x: i32) {
    println!("The value of x is: {x}");
}
```

Statements and Expressions

Statements are instructions that perform some action and do not return a value. Expressions evaluate to a resultant value.

statement

```
fn main() {
    let y = 6;
}
```

Functions, Comments

Expression

x+1 does not have a semicolon , if we add a semicolon it will become statement.

```
fn main() {
    let y = {
        let x = 3;
        x + 1
    };

    println!("The value of y is: {y}");
}
```

Functions with Return Values

must declare their type after an arrow (`->`).

final expression should equal to type of return value

You can return early from a function by using the return keyword with a value

```
fn five() -> i32 {
    5
}

fn main() {
    let x = five();

    println!("The value of x is: {x}");
}
```

Comments

Single line

```
// hello, world
```

Multi line

```
// So we're doing something complicated here, long enough that we need
// multiple lines of comments to do it! Whew! Hopefully, this comment will
// explain what's going on.
```