Functional Programming

Lecture 1

Komi Golova (she/her)
komi.golov@jetbrains.com

Constructor University Bremen

What is FP?

What is imperative programming?

Let's start simple:

```
auto x = 0;
x = 5;
```

What does this do?

What is imperative programming?

Let's start simple:

```
auto x = 0;
x = 5;
```

What does this do?

1. Create a "place" x.

What is imperative programming?

Let's start simple:

```
auto x = 0;
x = 5;
```

What does this do?

1. Create a "place" x.
2. Store the value 0 in x.

What is imperative programming?

Let's start simple:

```
auto x = 0;
x = 5;
```

What does this do?

1. Create a "place" x.
2. Store the value 0 in x.
3. Store the value 5 in x.

What is imperative programming?

Let's start simple:

```
auto x = 0;
x = 5;
```

What does this do?

1. Create a "place" x.
2. Store the value 0 in x.
3. Store the value 5 in x.

The place x is not the same as the value of x.

What is imperative programming?

Let's start simple:

```
auto x = 0;
x = 5;
```

What does this do?

1. Create a "place" x.
2. Store the value 0 in x.
3. Store the value 5 in x.

The place x is not the same as the value of x.

For example: `&x` is allowed, `&5` is not.

What is functional programming?

Let's go even simpler:

```
def x := 0
```

What does this do?

What is functional programming?

Let's go even simpler:

```
def x := 0
```

What does this do?

1. Define `x` to be another name for the value `0`.

What is functional programming?

Let's go even simpler:

```
def x := 0
```

What does this do?

1. Define `x` to be another name for the value `0`.

That's all!

- `x` is not a place.
- `x` does not store a value.
- Later, `x := 5` (i.e. `0 := 5`) does not make sense.

Functional programming is programming that focuses on *values*.

But... places!

FAQ: I like mutable variables! Can I still follow this course?

But... places!

FAQ: I like mutable variables! Can I still follow this course?

Creating and modifying places is a kind of *effect*.

But... places!

FAQ: I like mutable variables! Can I still follow this course?

Creating and modifying places is a kind of *effect*.

Other effects:

- Reading and writing input
- Picking a value non-deterministically
- Throwing an exception

But... places!

FAQ: I like mutable variables! Can I still follow this course?

Creating and modifying places is a kind of *effect*.

Other effects:

- Reading and writing input
- Picking a value non-deterministically
- Throwing an exception

Functional languages often support effects, but do so by building them on top of values. We will see how in this course.

But why?

FAQ: My day job will use Python. Why should I learn FP?

But why?

FAQ: My day job will use Python. Why should I learn FP?

Reason 1: Your Python code will still work with values.

But why?

FAQ: My day job will use Python. Why should I learn FP?

Reason 1: Your Python code will still work with values.

Reason 2: You will get a deeper understanding of your code.

But why?

FAQ: My day job will use Python. Why should I learn FP?

Reason 1: Your Python code will still work with values.

Reason 2: You will get a deeper understanding of your code.

Reason 3: Modern libraries focus on a functional style.

But why?

FAQ: My day job will use Python. Why should I learn FP?

Reason 1: Your Python code will still work with values.

Reason 2: You will get a deeper understanding of your code.

Reason 3: Modern libraries focus on a functional style.

Reason 4: Features from functional languages come to imperative ones.

Lean

What is Lean?

Lean 4 is a *functional programming language*
and *interactive theorem prover* developed by Microsoft.

What is Lean?

Lean 4 is a *functional programming language*
and *interactive theorem prover* developed by Microsoft.

Functional programming language: you can define programs that operate on values.

What is Lean?

Lean 4 is a *functional programming language*
and *interactive theorem prover* developed by Microsoft.

Functional programming language: you can define programs that operate on values.

Interactive theorem prover: you can reason about values and operations on those values.

What is Lean?

Lean 4 is a *functional programming language*
and *interactive theorem prover* developed by Microsoft.

Functional programming language: you can define programs that operate on values.

Interactive theorem prover: you can reason about values and operations on those values.

Together: you can reason about your programs!

What is Lean?

Lean 4 is a *functional programming language*
and *interactive theorem prover* developed by Microsoft.

Functional programming language: you can define programs that operate on values.

Interactive theorem prover: you can reason about values and operations on those values.

Together: you can reason about your programs!

Important note: we use Lean **4**.
Lean 3 is very different!

Who uses Lean?

Lean is primarily popular amongst mathematicians.

- Large amounts of mathematics is available in the Mathlib.
- LLMs + Lean have gotten gold on the IMO.

Who uses Lean?

Lean is primarily popular amongst mathematicians.

- Large amounts of mathematics is available in the Mathlib.
- LLMs + Lean have gotten gold on the IMO.

Lean is also good for cases when you need correctness.

- Amazon used Lean to develop Cedar.

Who uses Lean?

Lean is primarily popular amongst mathematicians.

- Large amounts of mathematics is available in the Mathlib.
- LLMs + Lean have gotten gold on the IMO.

Lean is also good for cases when you need correctness.

- Amazon used Lean to develop Cedar.

As AI becomes more common, being able to write specifications and automatically check them becomes more important.

Why Lean?

Wasn't this course supposed to be in Haskell? Well...

1. Lean is more user-friendly and interactive.
2. Lean is easier to setup.

Why Lean?

Wasn't this course supposed to be in Haskell? Well...

1. Lean is more user-friendly and interactive.
2. Lean is easier to setup.
3. Lean has features Haskell doesn't.
4. Lean is also useful for mathematics.

Why Lean?

Wasn't this course supposed to be in Haskell? Well…

1. Lean is more user-friendly and interactive.
2. Lean is easier to setup.
3. Lean has features Haskell doesn't.
4. Lean is also useful for mathematics.
5. Lean is growing in popularity.

Why Lean?

Wasn't this course supposed to be in Haskell? Well…

1. Lean is more user-friendly and interactive.
2. Lean is easier to setup.
3. Lean has features Haskell doesn't.
4. Lean is also useful for mathematics.
5. Lean is growing in popularity.
6. If you know Lean, you can learn Haskell/Agda/Idris/Arend/Rocq…

Learning goals and materials

Our goals:

1. Primary: teach you to program in a functional style (in Lean).
2. Secondary: teach you to reason about your code (in Lean).

Learning materials:

- Functional Programming in Lean
- Theorem Proving in Lean 4

Logistics

There are a lot of you!

This course was planned for about 30 students...

... but we have 80 registrations.

There are a lot of you!

This course was planned for about 30 students…

… but we have 80 registrations.

This has some consequences:

- Homework will be individual, via GitHub Classrooms.

There are a lot of you!

This course was planned for about 30 students…

… but we have 80 registrations.

This has some consequences:

- Homework will be individual, via GitHub Classrooms.
- All homework will be graded automatically.

There are a lot of you!

This course was planned for about 30 students…

… but we have 80 registrations.

This has some consequences:

- Homework will be individual, via GitHub Classrooms.
- All homework will be graded automatically.
  - Some manual checks to see you're not looking for exploits.
  - No partial credit for exercises (sorry).

There are a lot of you!

This course was planned for about 30 students…

… but we have 80 registrations.

This has some consequences:

- Homework will be individual, via GitHub Classrooms.
- All homework will be graded automatically.
  - Some manual checks to see you're not looking for exploits.
  - No partial credit for exercises (sorry).
  - Details about the grading will be available soon.

There are a lot of you!

This course was planned for about 30 students…

… but we have 80 registrations.

This has some consequences:

- Homework will be individual, via GitHub Classrooms.
- All homework will be graded automatically.
  - Some manual checks to see you're not looking for exploits.
  - No partial credit for exercises (sorry).
  - Details about the grading will be available soon.
- Help each other, discuss questions, ask online.

There are a lot of you!

This course was planned for about 30 students…

… but we have 80 registrations.

This has some consequences:

- Homework will be individual, via GitHub Classrooms.
- All homework will be graded automatically.
  - Some manual checks to see you're not looking for exploits.
  - No partial credit for exercises (sorry).
  - Details about the grading will be available soon.
- Help each other, discuss questions, ask online.
- Final exam will be on paper.

Telegram group (link)

Grading

Your grade consists of:

- 50% homework grade
  - Goal: help you understand the material.
- 50% exam grade
  - Goal: check your understanding.

You need at least 45% on **each part** to pass the course.

Tutorial sessions

After every lecture, there is an in-person tutorial session so that you can try the things you just saw.

Learning a language is mostly about practice!
This is not a spectator sport!

Tutorial sessions

After every lecture, there is an in-person tutorial session so that you can try the things you just saw.

Learning a language is mostly about practice!
This is not a spectator sport!

Your classamate Mikhail Vorobev (`@purely_injected` on Telegram) has kindly agreed to lead the tutorial sessions and answer your questions.

Communication

If you have questions:

1. Discuss it with a classmate.
2. Ask in the group.
   - We may split out a “help” group if needed.
3. Ask Mikhail (`@purely_injected`).
4. Ask me (`@jesyspa` or `komi.golov@jetbrains.com`).
5. Ask the Lean Zulip server.

Communication

If you have questions:

1. Discuss it with a classmate.
2. Ask in the group.
   - We may split out a “help” group if needed.
3. Ask Mikhail (`@purely_injected`).
4. Ask me (`@jesyspa` or `komi.golov@jetbrains.com`).
5. Ask the Lean Zulip server.

Questions so far?