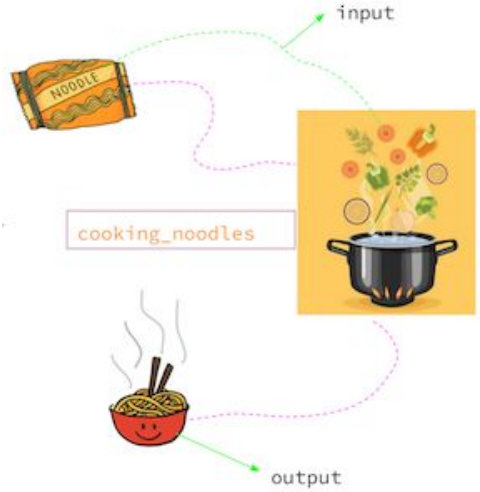
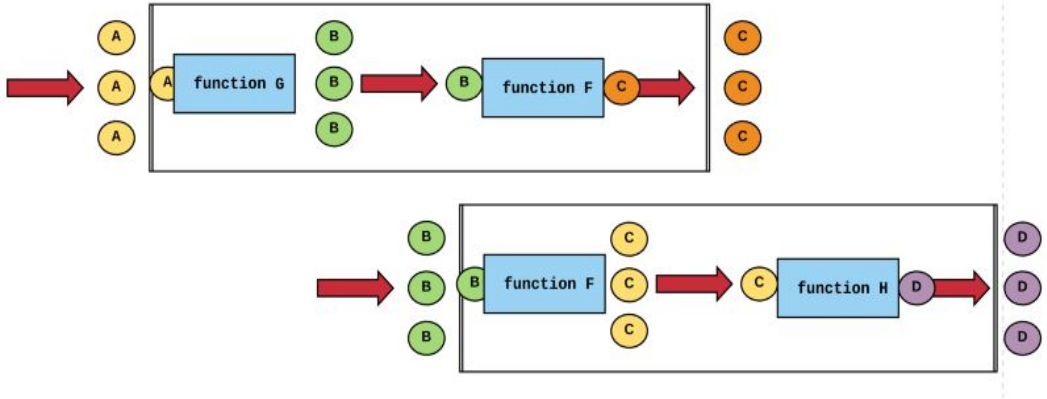
Functional Programming

Ashley, Harrison, Jenna, Jing, Parmanand

PROGRAMMING AS A RECIPE

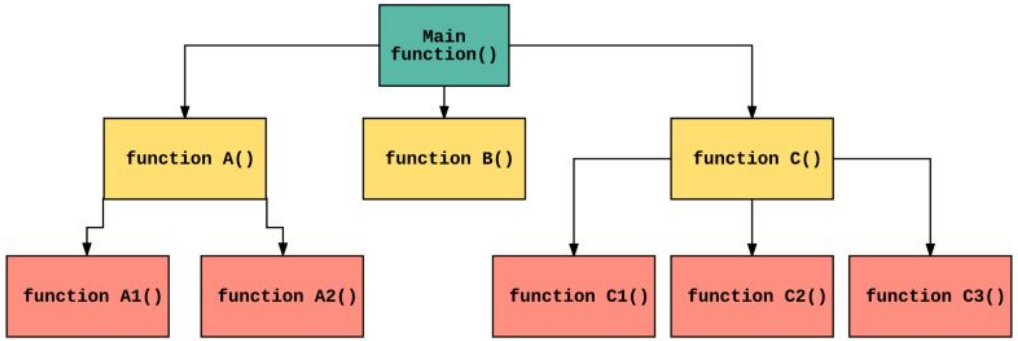


- Functional programming is like cooking with a recipe.
- Just as a recipe is a set of instructions that takes in inputs (ingredients) and produces output (a meal), functional programming is a set of functions that takes in arguments and produces a result.
- The functions in functional programming can be thought of as individual steps in a recipe, each performing a specific task to transform the input data into the desired output.



MIX & MATCH: RECIPE FOR EFFICIENT CODE

- Just as you can modify a recipe by adding or removing ingredients to create a new dish, you can modify a functional program by combining or removing functions to create a new program.
- Functional programming emphasize on breaking down a problem into smaller, more manageable functions, which can then be combined to solve the problem at hand. This approach enables developers to write code that is both efficient and easy to maintain.



THE KEY TO CLEAN AND MAINTAINABLE CODE

- As programs become larger, they also become more complex and difficult to understand, leading to potential errors.
- Functional programming is a programming paradigm that emphasizes the use of pure functions, immutability, and higher-order functions to create efficient and maintainable code.
- Functional programming emphasizes using functions to express computations, treating them as first-class citizens.
    - They can be assigned to variables, passed as arguments, to other functions, and return as values from functions.

BENEFITS OF FUNCTIONAL PROGRAMMING

1. **Increased Code Reusability**: functional programming encourages the creation of small, modular, and composable functions, which can be reused across different parts of a program or even different programs.
2. **Easier Debugging**: functional programming emphasizes pure functions, immutability, and separation of concerns, which make it easier to reason about the behavior of the code and identify and fix errors.
3. **Improved Concurrency Support**: functional programming encourages the use of immutable data structures and pure functions, which are well-suited for parallel and distributed computing environments.
4. **Better Reliability:** functional programming reduces the potential for side effects and makes it easier to handle errors, leading to more reliable software.
5. **Improved Maintainability**: functional programming encourages modularity and separation of concerns, making it easier to maintain and update code over time.
6. **Increased Developer Productivity**: functional programming can lead to shorter development cycles and faster time-to-market due to its focus on code reuse and expressiveness.

Object-Oriented Programming

- Used for executing fewer operations with common behavior and different variants.
- Makes use of mutable data for functioning.
- Preferred when there are many inputs to process but with fewer operations to perform
- Conditional statements can be used: switch & if-else
- An object is the primary manipulation unit

Functional Programming

- Used for executing many different operations for which the data is fixed.
- Works well with immutable data
- Preferred when there is a requirement of more operations and only a few inputs
- Doesn’t support conditional statements.
- Mainly based on the usage of functions and variables.

FUNCTIONAL PROGRAMMING IN ACTION

Functional programming has gained popularity in both industry and academia due to its benefits in creating efficient, reliable, and maintainable code.

Companies that uses functional programming include Facebook (Haskell), Twitter (Scala), Jane Street (OCaml), and WhatsApp (Erlang).

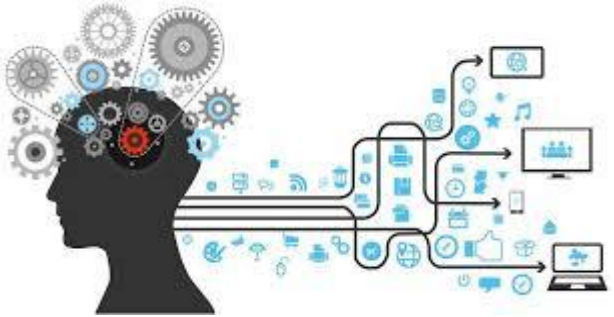
Functional programming is also widely used in academic research, particularly in the fields of computer science and mathematics.

FUNCTIONAL PROGRAMMING IN ACTION

Web Development

Functional programming is commonly used in web development for building server-side applications and APIs using languages such as Haskell and Erlang.

AI and Machine Learning



Functional programming is used in artificial intelligence and machine learning applications to process and analyze large datasets.

FUNCTIONAL PROGRAMMING IN ACTION

Game Development

Functional programming is used in the gaming industry for developing game engines and designing game logic.

Financial Modeling

Functional programming is used in financial applications for modeling complex financial systems, pricing models, and risk analysis.

FUNCTIONAL PROGRAMMING LANGUAGES

Haskell

a statically typed functional programming language that is purely functional and has lazy evaluation. Advantages include strong type safety and good performance, while disadvantages include a steep learning curve and a relatively small community.

Scala

a hybrid functional programming language that is compatible with Java and runs on the JVM. Advantages include interoperability with Java and a strong focus on object-oriented programming, while disadvantages include some complexity and a less pure approach to functional programming.

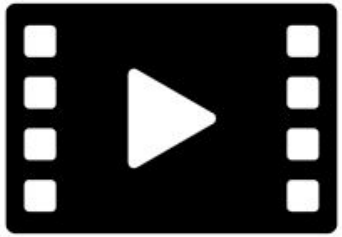
Racket

a multi-paradigm programming language that supports both functional and object-oriented programming. Advantages include good support for macros and a large standard library, while disadvantages include a relatively small community and some performance issues.

Note: There are many other functional programming languages available, and this is not an exhaustive list. The advantages and disadvantages mentioned are just a few examples and may vary depending on specific use cases and personal preferences.

INSTALLING DRRACKET

Instructions

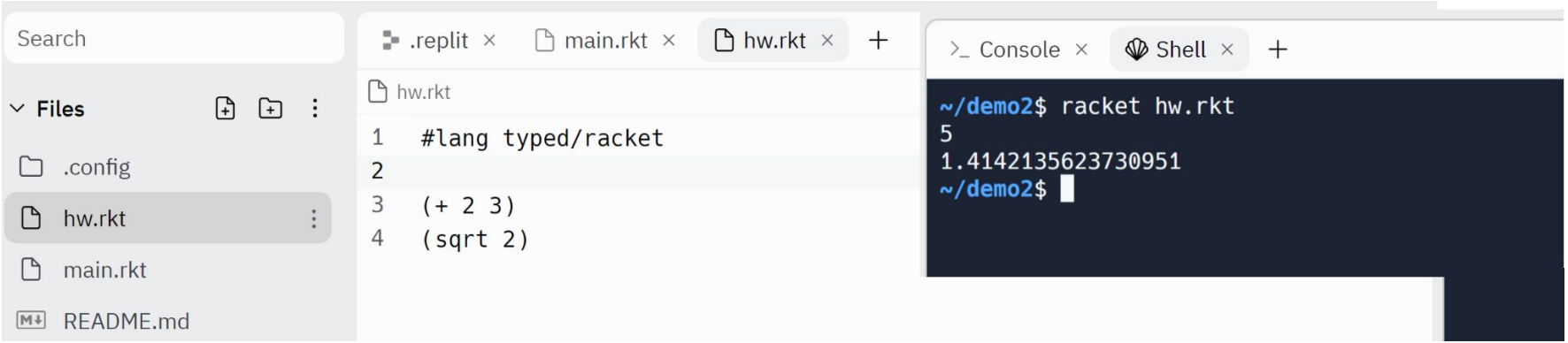


Video Instructions

(Mac Window)

IF YOU PREFER TO CODE ON REPLIT INSTEAD!

- In replit, create new and type in racket in the template
- You may click the green “run” button to run the main.rkt in console
- If you want to run a new file (hw.rkt), make sure you add the line, “#lang typed/racket”, at the beginning of the new file. Then you may type “racket hw.rkt” in the shell to run it.



INTRO TO RACKET PROGRAMMING

Instructions:

Please watch this 4-min video introducing DrRacket. The video explains how to write and run code on DrRacket, use comments, and define variables.

DEFINITION, FUNCTIONS, CONDITIONS

Instructions:

Please watch this 10-min video on Racket basics, covering concepts like defining functions, variables, and conditionals.

ADDITIONAL RACKET RESOURCES (OPTIONAL)

If you are interested in learning more about Racket:

- The Racket Guide
- Coursera: Introduction to Racket Programming

Other Intro to Racket Videos:

- An Introduction to the Racket Programming Language
- Racket Programming - Introduction to Racket: calling and defining functions, if, cond.
- Racket: List functions: first, rest, null?, list, append, cons

Glossary

Pure Function

Click the title for a video explanation.

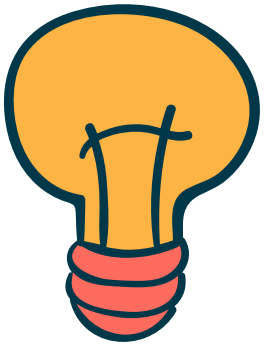
Pure functions do not have side effects meaning that the function will always produce a given output for a certain input. They are similar to mathematical functions, because they will always return the same value. If there were side-effects, a function would have an effect on them and then return it.

Immutability

Click the title for a video explanation.

Since there is no state in functional programming, we are working with immutable data. This means that the functions that do the work with the given instructions will have **predictable** outcomes. For example, say that you are lactose intolerant. No matter how many times you try to eat a piece of pizza, you still might get sick. It is easy to predict that you will get sick.

There is only the binding of names to values which is referred to as value semantics. This can be seen as restrictive since the values never change once they enter the environment.



State

Click the title for a video explanation.

State in programming could be better explained with an example of what State is outside of programming. For example, if I were drinking a cup of coffee, you could say that my state is that I am drinking a cup of coffee. However, having my mouth open to drink the coffee is another state of drinking coffee. So is the motion of moving my arms to my mouth.

There are therefore multiple states that drinking coffee has. All of these different states when combined together, make up the program. Therefore, another term for state is “data”, as they describe what the program is doing.

Higher-Order Functions

Click the title for a video explanation.

Functions that take one or more functions as input parameters and return a function as output.