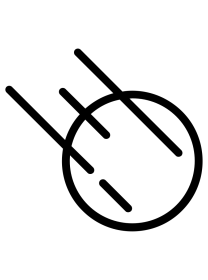
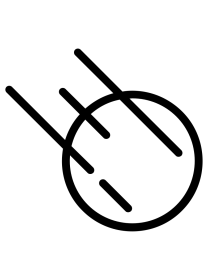


Welcome – CSC493

Introduction to Multi-Paradigm Programming

- Instructor: Dr Lutz Hamel
- Email: lutzhamel@uri.edu

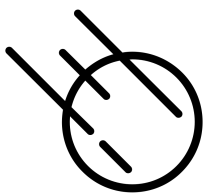
- Website: lutzhamel.github.io/CSC493/
- BrightSpace



What is a Paradigm?

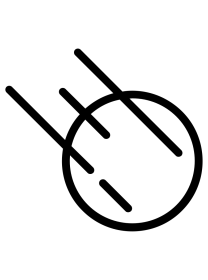
- A paradigm is a distinct set of concepts and practices that define a discipline.

Source: https://en.wikipedia.org/wiki/Paradigm



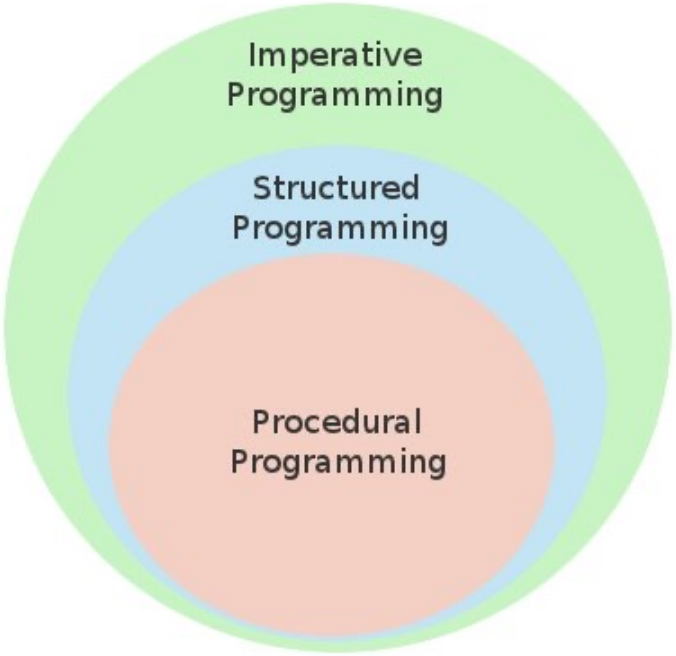
What is a Programming Paradigm?

- A programming paradigm is an approach to programming using a distinct set of concepts and practices. E.g.
  - **Imperative programming paradigm** – explicit statements that change the program state
  - **Object-oriented programming paradigm** – uses data structures consisting of data fields and methods together with their instantiations (objects) to design programs
  - **Functional programming paradigm** – uses evaluation of mathematical functions where everything is considered a value and avoids explicit state manipulation
  - **Pattern matching paradigm** – uses patterns to access or destructure data structures (declarative programming).

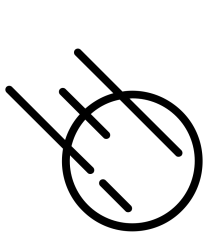


Imperative Programming

- There is a lot of confusion of terminology around imperative, structured, and procedural programming.
- However, these terms form roughly a hierarchy as seen on the right.
- When we talk about imperative programming we mean all these things



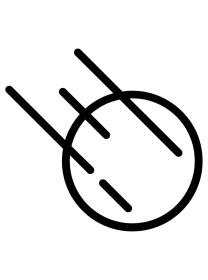
Note: there are many exceptions, e.g., you can have procedural programs with goto statements – not structured! Think C.



What is a Multi-Paradigm Programming Language?

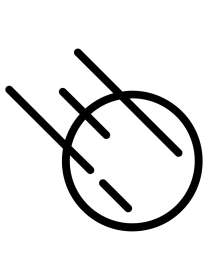
- A multi-paradigm programming language is a programming language supporting more than one programming paradigm, in order to allow the most suitable programming style for a task.

Source: https://en.wiktionary.org/wiki/multi-paradigm



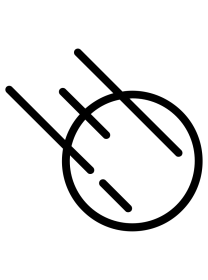
Why Study?

- Different programming paradigms provide different tools/approaches to tackle programming challenges
- Picking the right paradigm for the job at hand is an essential skill of every software developer



Our Languages

- The languages we will be discussing/using all support the following to varying degrees,
  - Imperative programming
  - OOP
  - Functional programming
  - Pattern matching

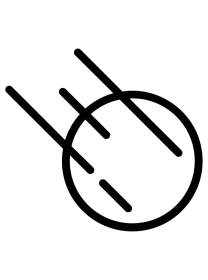


Python

- www.python.org/

```
# hello world as a Python program
print("Hello, World!")
```

ln001/hello.py

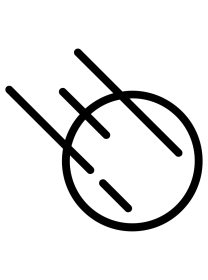


Rust

- www.rust-lang.org/

```rust
// hello world as a Rust program
fn main() {
    println!("Hello, World!");
}
```

ln001/hello.rs

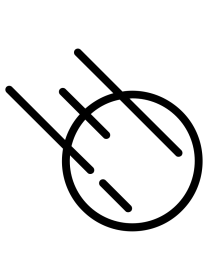


Asteroid

- asteroid-lang.org/

```
-- hello world as an Asteroid program
load system io.
io @println "Hello, World!".
```

ln001/hello.ast



Reading

- Installing and Running Asteroid
  - asteroid-lang.readthedocs.io/en/latest/Installing%20and%20Running.html
- Intro
  - asteroid-lang.readthedocs.io/en/latest/User Guide.html#introduction