What this course is NOT about

- Learning to program in [insert language here]
  - but this course should make it easier to learn new languages

“Language wars” (“Which is better, Java or C++?”)
- but you will learn about criteria that can be used to compare different languages
What this course is NOT about

- Learning to program in [insert language here]
  - but this course should make it easier to learn new languages

- Learning tiny bits about lots of different languages
  - but we will use examples from many languages to examine more general principles
What this course is NOT about

- Learning to program in [insert language here]
  - but this course should make it easier to learn new languages

- Learning tiny bits about lots of different languages
  - but we will use examples from many languages to examine more general principles

- “Language wars” (“Which is better, Java or C++?”)
  - but you will learn about criteria that can be used to compare different languages
What this course is about

- How are languages designed and implemented?
  - Specifying syntax and semantics, compiled vs. interpreted, etc.

- How do such choices affect ease of use, efficiency, scalability, and other criteria?
  - Example: How should recursive calls be implemented? How does "garbage collection" work?

- What are the different programming paradigms?
  - Example: Why would anyone ever use a language like ML rather than C++ or Java?
What this course is about

- How are languages designed and implemented?
  - Specifying syntax and semantics, compiled vs. interpreted, etc.

- How do such choices affect ease of use, efficiency, scalability, and other criteria?
  - Example: How should recursive calls be implemented? How does “garbage collection” work?
What this course is about

- How are languages designed and implemented?
  - Specifying syntax and semantics, compiled vs. interpreted, etc.
- How do such choices affect ease of use, efficiency, scalability, and other criteria?
  - Example: How should recursive calls be implemented? How does “garbage collection” work?
- What are the different programming paradigms?
  - Example: Why would anyone ever use a language like ML rather than C++ or Java?
Why Study Programming Languages?

- Help you choose a language.
- Make it easier to learn new languages.
- Help you make better use of whatever language you use.

Image credit: https://pixabay.com

Janyl Jumadinova

Programming Languages

September 1-3, 2020
Why Study Programming Languages?

- Help you choose a language.
Why Study Programming Languages?

- Help you choose a language.
- Make it easier to learn new languages.
Why Study Programming Languages?

- Help you choose a language.
- Make it easier to learn new languages.
- Help you make better use of whatever language you use.

Image credit: https://pixabay.com
Programming Language Paradigms

Paradigms

Imperative
- Procedural
- Object-Oriented
- Parallel Processing

Declarative
- Logic
- Functional
- Dataflow
- Database
Programming Language Paradigms

- **Declarative**
  - functional
  - dataflow
  - logic, constraint-based
  - template-based

- **Imperative**
  - von Neumann
  - scripting
  - object-oriented

- **Logic**
  - Lisp/Scheme, ML, Haskell
  - Prolog, spreadsheets
  - XSLT

- **Functional**
  - Id, Val
  - C, Ada, Fortran, ...
  - Perl, Python, PHP, ...

- **Database**
  - Smalltalk, Eiffel, Java, ...

Programming Language Paradigms

**Imperative**
- Procedural
- Object-Oriented
- Parallel Processing

**Declarative**
- Logic
- Functional
- Database

**Declarative**
- functional
- dataflow
- logic, constraint-based
- template-based

**Imperative**
- von Neumann
- scripting
- object-oriented

Lisp/Scheme, ML, Haskell
Id, Val
Prolog, spreadsheets
XSLT
C, Ada, Fortran, ...
Perl, Python, PHP, ...
Smalltalk, Eiffel, Java, ...

Image credits: https://thepafhelper.blogspot.com and PLP book
What makes a language successful?
Language Evaluation Criteria

- **Readability**: the ease with which programs can be read and understood
Language Evaluation Criteria

- **Readability**: the ease with which programs can be read and understood
- **Writability**: the ease with which a language can be used to create programs
Language Evaluation Criteria

- **Readability**: the ease with which programs can be read and understood
- **Writability**: the ease with which a language can be used to create programs
- **Reliability**: conformance to specifications
Language Evaluation Criteria

- **Readability**: the ease with which programs can be read and understood
- **Writability**: the ease with which a language can be used to create programs
- **Reliability**: conformance to specifications
- **Cost**: the ultimate total cost
Art of Programming Language Design Activity

"...design [is] the intentional creation of plans for a new kind of thing". Parsons
"...design [is] the intentional creation of plans for a new kind of thing". Parsons

- Choose one programming language
- Working in a group, investigate the following:
  1. Key characteristics of the language **design**.
  2. Historical context (how it started, when, by whom, the first language, etc.).
  3. The purpose/usage (why is it there).
- Prepare to share your findings.

Image credit: https://pixabay.com

Janyl Jumadinova
Programming Languages
Why So Many Programming Languages?

Evolution
Socio-economic factors
Special purposes
Special hardware
Range of ideas
Why So Many Programming Languages?

- Evolution
Why So Many Programming Languages?

- Evolution
- Socio-economic factors
Why So Many Programming Languages?

- Evolution
- Socio-economic factors
- Special purposes
Why So Many Programming Languages?

- Evolution
- Socio-economic factors
- Special purposes
- Special hardware
Why So Many Programming Languages?

- Evolution
- Socio-economic factors
- Special purposes
- Special hardware
- Range of ideas
Before our class next week, make sure you have Docker set up and working. Check out department’s video (on Slack) overviewing its setup.