CS 220: Survey of Programming Languages

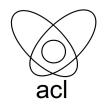
Lecture Slides

Introduction

Jan Michael C. Yap

Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines, Diliman
jcyap@dcs.upd.edu.ph

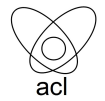
Session 1



Introduction

Programming Languages

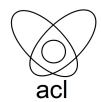
Programming Language Paradigms



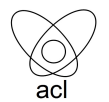
Introduction

Programming Languages

Programming Language Paradigms



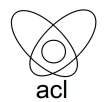
Software tools



Software tools

- In general, software tools can be studied as interfaces between clients and lower-level facilities.

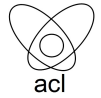


Software tools

- In general, software tools can be studied as interfaces between clients and lower-level facilities.
- Some important questions

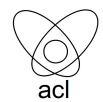


Software tools

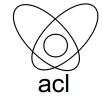
- In general, software tools can be studied as interfaces between clients and lower-level facilities.
- Some important questions
  - What is the nature of the interface?



Software tools

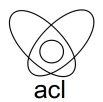
- In general, software tools can be studied as interfaces between clients and lower-level facilities.
- Some important questions
  - What is the nature of the interface?
  - How can the interface be implemented by using the lower-level facilities?



Software tools

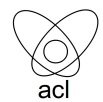
- In general, software tools can be studied as interfaces between clients and lower-level facilities.
- Some important questions
  - What is the nature of the interface?
  - How can the interface be implemented by using the lower-level facilities?
  - How useful is the interface for humans or their agents?



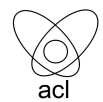
Programming languages as software tools

- Translating those questions into discussion of programming languages...



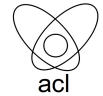
Programming languages as software tools

- Translating those questions into discussion of programming languages...
  - What is the structure (syntax) and meaning (semantics) of the programming language constructs?



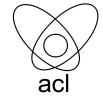
Programming languages as software tools

- Translating those questions into discussion of programming languages...
  - What is the structure (syntax) and meaning (semantics) of the programming language constructs?
  - How does the compiler deal with the constructs in order to translate them into assembler or machine language?

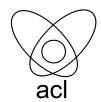


Programming languages as software tools

- Translating those questions into discussion of programming languages...
  - What is the structure (syntax) and meaning (semantics) of the programming language constructs?
  - How does the compiler deal with the constructs in order to translate them into assembler or machine language?
  - Is the programming language easy to use, expressive, readable, and protects the programmer from programming errors?

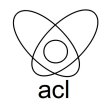


Introduction



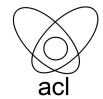
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.



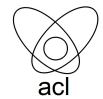
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course



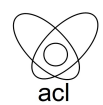
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course
  - Imperative



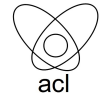
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course
  - Imperative
  - Object-Oriented



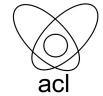
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course
  - Imperative
  - Object-Oriented
  - Functional



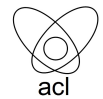
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course
  - Imperative
  - Object-Oriented
  - Functional
  - Dataflow



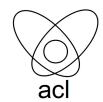
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course
  - Imperative
  - Object-Oriented
  - Functional
  - Dataflow
  - Concurrent



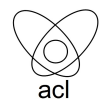
Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course
  - Imperative
  - Object-Oriented
  - Functional
  - Dataflow
  - Concurrent
  - Logic



Programming language paradigms

- Different paradigms impose very different programming styles, but even more important, they change the way the programmer looks at algorithms.
- Covered paradigms in the course
  - Imperative
  - Object-Oriented
  - Functional
  - Dataflow
  - Concurrent
  - Logic
  - Aggregate



END OF SESSION 1