CS-210  
Functional programming  
Kuncak Viktor, Odersky Martin

<table>
<thead>
<tr>
<th>Cursus</th>
<th>Sem.</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HES - IN</td>
<td>H</td>
<td>Obl.</td>
</tr>
<tr>
<td>Informatique</td>
<td>BA3</td>
<td>Obl.</td>
</tr>
<tr>
<td>Systèmes de communication</td>
<td>BA3</td>
<td>Opt.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Language</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credits</td>
<td>5</td>
</tr>
<tr>
<td>Session</td>
<td>Winter</td>
</tr>
<tr>
<td>Semester</td>
<td>Fall</td>
</tr>
<tr>
<td>Exam</td>
<td>During the semester</td>
</tr>
<tr>
<td>Workload</td>
<td>150h</td>
</tr>
<tr>
<td>Weeks</td>
<td>14</td>
</tr>
<tr>
<td>Hours</td>
<td>4 weekly</td>
</tr>
<tr>
<td>Lecture</td>
<td>2 weekly</td>
</tr>
<tr>
<td>Exercises</td>
<td>2 weekly</td>
</tr>
</tbody>
</table>

Summary
Understanding of the principles and applications of declarative programming, the fundamental models of program execution, application of fundamental methods of program composition, meta-programming through the construction of interpreters and advanced programming techniques.

Content
Introduction to programming in Scala  
Expressions and functions  
Classes and objects  
Evaluation by rewriting  
Pattern matching  
Polymorphism  
Evaluation strategies  
Domain-specific languages  
Constraint programming  
Language interpretation  
An interpreter for Lisp  
An interpreter for Prolog

Learning Prerequisites
Required courses
Introduction to the programming objet  
Theory and practice of programming

Important concepts to start the course
Compiler Construction  
Foundations of Software

Learning Outcomes
By the end of the course, the student must be able to:
• Create functional programs  
• Design robust and readable software  
• Formalize program correctness  
• Interpret programs automatically
• Prove correctness using induction
• Construct software

Transversal skills
• Demonstrate a capacity for creativity.
• Use a work methodology appropriate to the task.
• Set objectives and design an action plan to reach those objectives.
• Give feedback (critique) in an appropriate fashion.

Teaching methods
MOOC. Ex Cathedra. Exercises and projects

Assessment methods
Continuous and written test at the end of the course

Resources
Bibliography
Abelson/Sussman : Structure and Interpretation of Computer Programs, MIT Press

Ressources en bibliothèque
• Structure and Interpretation of Computer Programs / Abelson

Websites
• http://Lampwww.epfl.ch/teaching