

**Module title: Programming Paradigms**

Module ID	Workload	Credits	Semester	Frequency of Offering	Duration
MI11	150 h	5	1	yearly	1 semester

Workload	Attendance	Preparation and Follow-Up	Private Study	Preparation for Exam and Exam	Total
S	30 h / 2 SWS	15 h	45 h	10 h	
P	15 h / 1 SWS	45 h			
Total	45 h / 3 SWS	60 h	45 h	10 h	<b>150 h</b>

**1** **Scheduled Group Size:** S: 15 students, P: 15 students

**2** **Subject Knowledge / Skills**

*The objective target of this module is the acquisition of basic knowledge of the most prominent current programming paradigms. In addition to substantial basic knowledge in imperative and object oriented programming, students will gain a good understanding in the areas of functional and logic programming.*

*The students are able to compare different approaches to programming and are able to identify and name commonalities and differences between programming languages. They know and understand the conditions under which a specific programming paradigm is applicable and are able to specify the limits of at least four different programming paradigms. They have gained insight into the historical development of programming languages and are able to recognise these concepts in new programming languages.*

**3** **Content / Syllabus**

*Introduction to programming and programming languages*

*A short history of programming languages*

*The concept of infinity*

*A short introduction to the theory of computation*

*Imperative programming*

*Modular programming with functions and procedures*

*Divide and Conquer as a basic programming concept*

*Example languages: Python*

*Object oriented programming*

*Basic concepts: classes, objects, inheritance, polymorphism*

*Programming with interfaces*

*Aggregation and composition*

*Design patterns*

*Example languages: C# and Java*

*Functional programming*

*Mathematical Notation: Lambda calculus and currying*

*Higher order functions*

*Comprehension and memoization*

*Example languages: F# and Scheme*

	<p><i>Logic programming</i>  <i>Logic notation using Horn clauses</i>  <i>Unification as a basic reasoning mechanism</i></p>
<b>4</b>	<p><b>Teaching Format</b></p> <p><i>Seminar accompanied by practical work in a laboratory environment</i></p>
<b>5</b>	<p><b>Prerequisites</b></p> <p><i>None</i></p>
<b>6</b>	<p><b>Recommended Qualifications for the Participation</b></p> <p><i>None</i></p>
<b>7</b>	<p><b>Assessment</b></p> <p><i>Written exam</i></p>
<b>8</b>	<p><b>Prerequisites for Granting ECTS Credits</b></p> <p><i>Exam passed</i></p>
<b>9</b>	<p><b>Usage of this Module in Other Degree Courses</b></p> <p><i>Master Applied Computer Science</i></p>
<b>10</b>	<p><b>Contribution to Final Score</b></p> <p>5,56 %</p>
<b>11</b>	<p><b>Convenor</b></p> <p>Professor of Digital Media Computing and Web Technologies</p>
<b>12</b>	<p><b>Language of Instruction</b></p> <p>English</p>
<b>13</b>	<p><b>Reading List</b></p> <ul style="list-style-type: none"> <li>• Vivek Kulkarni. <i>Theory of Computation</i>, Oxford University Press, 2013</li> <li>• Mark Lutz. <i>Learning Python</i>, O'Reilly &amp; Associates, 2013.</li> <li>• Bishop, Judith. <i>C# 3.0 Design Patterns</i>, O'Reilly Media, 2008.</li> <li>• Tomas Petricek, Jon Skeet. <i>Real-World Functional Programming: With Examples in F# and C#</i>, Manning Publications, 2010.</li> <li>• Ehud Shapiro. <i>The Art of Prolog: Advanced Programming Techniques (Logic programming)</i>, MIT Press, 1994.</li> </ul>