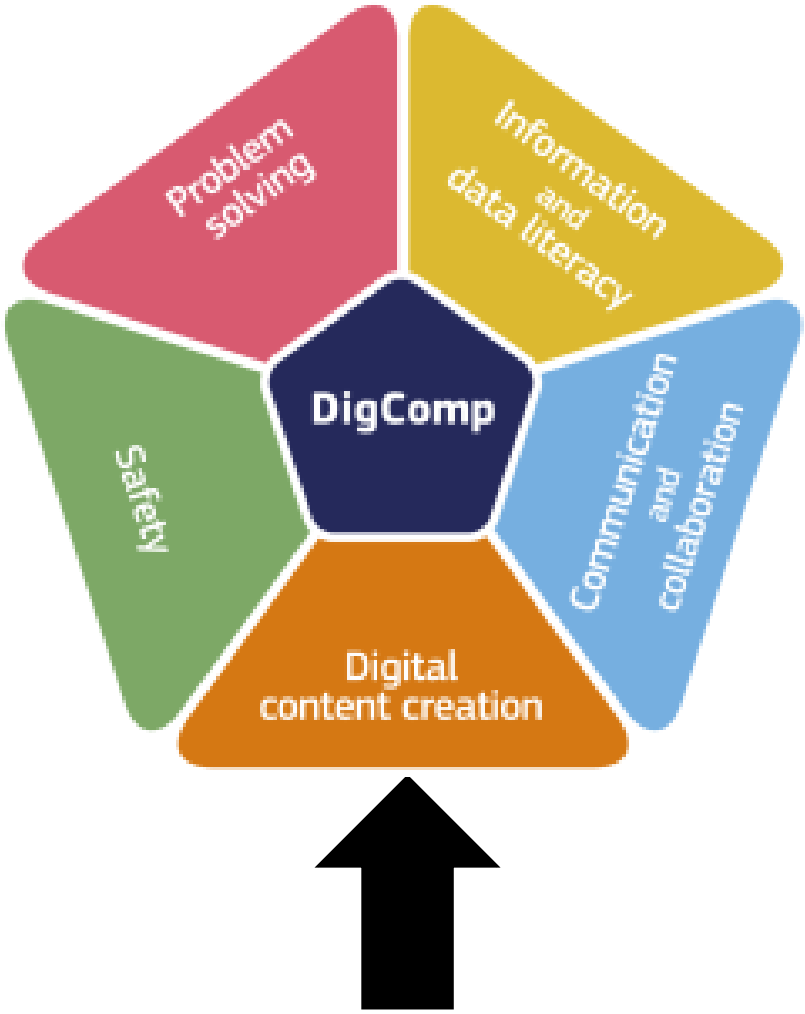
MICROCREDENTIALS FOR DIGITAL CONTENT CREATION

Competence 3.4: PROGRAMMING

DSW
DIGITAL SKILLS WALLET

Co-funded by
the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.


Problem solving
Information and data literacy
DigComp
Communication and collaboration
Safety
Digital content creation

Content

FOUNDATION LEVEL

(Level 1 and Level 2)

Programming languages – types and classification (MC 3.4.A.1)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Programming languages – types and classification Code: MC **3.4.A.1**
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	FOUNDATION
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.A.1)

Learning Outcomes (ref. Level 1-2 LOs 3.4.1 - 3.2.4):

- Understanding types of application that can be developed into a programming language.
- Explain the characteristics of different types of applications regarding working platform and environment.
- Is aware on how the programming language can be used to develop applications for various systems and environments.
- Enumerate a list with common language programming.

Description (MC 3.4.A.1)

At this foundational level of expertise in programming within digital content creation, developers should comprehend the diverse application types developed through programming languages. They should have a foundational understanding that desktop applications run on operating systems like Windows, macOS, or Linux, offering standalone functionality. Also, they should understand that mobile applications target platforms such as Android and iOS, tailoring to handheld devices. They should realize that web applications are used for website and service creation, accessible across various browsers. They should know that embedded applications involve specialized programming for dedicated functions in devices like IoT systems, ensuring compatibility with specific computing environments.

The developers should enumerate the characteristics of different application types recognizing key traits of application types based on their working platform (desktop, mobile, web, embedded), and environment (standalone software, optimized for touch interfaces, accessed through the internet, targeted for specific functions in devices requiring efficient resource utilization and real-time processing), recognizing the difference between platform-specific and cross-platform programming languages.

At the foundational level, individuals should be familiar with examples of programming languages, including C, C++, C#, Java, Kotlin, JavaScript, HTML, XML, React, Python, Ruby, Prolog, Lisp, and Scala.

Questions (MC 3.4.A.1)

1. Can you enumerate the main types of applications based on their implementation platform (desktop, mobile, web, embedded)?
2. Briefly describe the differences in how applications are developed for various systems and environments.
3. What are the key characteristics of desktop applications, particularly those running on operating systems like Windows, macOS, or Linux?
4. Explain the characteristics of mobile applications designed for platforms such as Android and iOS.
5. Describe how programming languages can be utilized to create web applications, websites, and web services.
6. How can a programming language be effectively used in programming embedded systems?
7. Give examples of programming languages for desktop/mobile/web/embedded applications.

IDE - Integrated Development Environment (MC 3.4.A.2)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	IDE - Integrated Development Environment Code: MC **3.4.A.2**
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	FOUNDATION
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.A.2)

Learning Outcomes (ref. Level 1-2 LOs 3.4.5 - 3.2.6):

- Identify what is an IDE – Integrated Development Environment.
- Is aware of the main features and functionalities for different IDEs.

Description (MC 3.4.A.2)

At the foundational level of expertise in programming, individuals should recognize IDE as acronym for "Integrated Development Environment" and understand its role in programming tasks related to digital content creation.

Developers should display awareness of the main features and functionalities of different IDEs. They comprehend the diverse tools and capabilities offered by various IDEs, demonstrating a capacity to navigate and utilize these tools effectively. This awareness includes familiarity with features tailored to digital content creation, ensuring efficiency in coding, debugging, and testing activities. These main features commonly found in an IDE are Code Editor, Compiler/Interpreter, Debugger, Project Management, Version Control, Build Automation, Code Templates and Snippets, Code Refactoring, Code Analysis for potential errors, Autocomplete, Error and Warning Highlighting tools, Find and Replace tools, Code Navigation, Documentation or external API references integration, Unit Testing Integration, Customization and Extensions.

Questions (MC 3.4.A.2)

1. What does IDE stand for, and how would you define it?
2. Explain IDE role in facilitating software development.
3. Can you identify and briefly describe three main features commonly found in an Integrated Development Environment (IDE) that contribute to enhancing a developer's productivity and coding experience?
4. What specific tools and functionalities does it provide for developers to identify and resolve bugs during the code execution process?
5. How does built-in support for version control systems, such as Git, benefit developers in managing code changes and collaborating with others?
6. How does autocomplete functionality in an IDE contribute to efficient coding and error reduction?

Choose and install an IDE (MC 3.4.A.3)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Choose and install an IDE. Code: MC **3.4.A.3**
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	FOUNDATION
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.A.3)

Learning Outcomes (ref. Level 1-2 LOs 3.4.7 - 3.2.8):

- Find and choose an IDE corresponding with a programming language.
- Install an IDE.

Description (MC 3.4.A.3)

At the foundation level of expertise in programming for digital content creation, programmers should possess practical skills related to selecting and utilizing Integrated Development Environments (IDEs).

Firstly, they should demonstrate the ability to find and choose an IDE suitable for a specific programming language, considering factors like language support, features, and compatibility. This reflects a fundamental understanding of the importance of IDEs in the development process.

Secondly, they should exhibit proficiency in installing an IDE, showcasing the practical skill of setting up the chosen development environment. This includes navigating installation processes, configuring settings, ensuring a continuous setup, and managing updates. This might involve configuring build tools, setting compilation options, and understanding the build process.

Questions (MC 3.4.A.3)

1. How would you find a list of available Integrated Development Environments (IDEs) suitable for a specific programming language on the Internet?
2. Given a set of characteristics and features associated with different IDEs, can you explain your decision-making process in choosing an IDE for a particular programming language?
3. Highlight at least three key characteristics you would prioritize on choosing an IDE.
4. Describe the steps involved in installing an IDE with predefined or custom settings. Include considerations such as downloading and running an installer, configuring settings, and managing updates.
5. How would you approach configuring build tools, setting compilation options, and understanding the build process within an IDE?
6. When installing an IDE with custom settings, how would you tailor the configuration to meet specific requirements?

Programming documentation (MC 3.4.A.4)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Programming documentation Code: MC **3.2.A.4**
Country(ies)/Region(s) of the issuer IRELAND,	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	FOUNDATION
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.A.4)

Learning Outcomes (ref. Level 1-2 LOs 3.4.9 - 3.2.11):

- Explain and find language programming documentation.
- Is able to run a simple program.

Description (MC 3.4.A.4)

At the foundation level of expertise in programming for digital content creation, developers should demonstrate proficiency in understanding and locating programming language documentation. They should recognize four key types: official language documentation, tutorials, posts/discussions, and reference materials. This foundational knowledge underscores the importance of diverse resources for effective learning and problem-solving.

Secondly, peoples should showcase practical skills in running a simple program inspired from documentation. They should create, edit, and save source code files within an Integrated Development Environment (IDE), understanding the specific file extensions. Additionally, they are able to build and compile code, often referencing official documentation for guidance. These skills reflect a hands-on approach to coding within a digital content creation context, laying the groundwork for more advanced programming tasks.

Questions (MC 3.4.A.4)

1. Explain the four types of programming language documentation.
2. Describe the process of creating, editing, and saving source code files within an Integrated Development Environment (IDE).
3. How do you ensure that the files have the correct extensions for the programming language you are working with?
4. In what ways do you leverage programming language documentation, particularly official documentation, when working on a coding project?
5. If you encounter an error during the code building and compilation process, how would you approach troubleshooting and resolving the issue?

Application domains (MC 3.4.A.5)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Application domains. Code: MC **3.2.A.5**
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	FOUNDATION
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.A.5)

Learning Outcomes (ref. Level 1-2 LOs 3.4.12 - 3.2.13):

- Classify the application domains.
- Identify specific demands and constraints for a specific application domain.

Description (MC 3.4.A.5)

At the foundation level of expertise in programming for digital content creation, developers should be able to classify application domains, understanding them as real-life areas where applications are intended to be applied. This foundational knowledge ensures an awareness of the diverse contexts in which digital content creation applications operate, such as finance, healthcare, education, e-commerce, transportation and logistics, entertainment and media, manufacturing and industrial automation, government, public services, IoT (Internet of Things), sciences.

Secondly, programmers showcase the ability to identify specific demands and constraints within a given application domain. They can pose thoughtful questions about critical considerations like user requirements, data structures, algorithms, security measures, and compliance standards. This skill reflects a capacity to analyze and comprehend the unique challenges associated with programming within a specific context, laying the groundwork for effective and targeted digital content creation.

Questions (MC 3.4.A.5)

1. Define the term "application domain" (or problem domain) in the context of software development.
2. Provide five examples of different application domains in software development.
3. How does recognizing the scope and context of an application domain influence the design and implementation of software?
4. Explain the role of the application domain in guiding developers during the software development process.
5. List and briefly describe key considerations a developer should address when working within a specific application domain.
6. Provide examples of software applications in at least three different application domains.

INTERMEDIATE LEVEL

(Level 3 and Level 4)

Specifications/functionalities of an application (MC 3.4.B.1)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Specifications/functionalities of an application Code: MC **3.2.B.1**
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.1)

Learning Outcomes (ref. Level 3-4 LOs 3.4.14 - 3.2.15):

- Explain the main specifications/functionalities for an application into a particular domain.
- Establish the main specifications/functionalities for an application into a particular domain.

Description (MC 3.4.B.1)

At the intermediate level of expertise in programming for digital content creation, developers should be able to explain the main specifications and functionalities required for an application within a specific domain. This involves understanding that domain-specific specifications are the unique requirements and capabilities essential for effectively addressing the needs of that domain. For instance, an e-commerce app requires user registration, a product catalog, and secure payment processing.

Secondly, peoples should exhibit the ability to establish these specifications by asking critical questions about future functions and constraints. They can inquire about user registration details, quiz retake options, and feedback mechanisms, ensuring a comprehensive understanding of the application's functionalities. This foundational skill set equips developers to align digital content creation with the distinctive characteristics and demands of diverse application domains, such as finance, healthcare, education, e-commerce, transportation and logistics, entertainment and media, manufacturing and industrial automation, government, public services, IoT (Internet of Things), sciences.

Questions (MC 3.4.B.1)

1. How do the specifications differ across various application domains, and why is it crucial for functionalities to align with domain characteristics?
2. Provide examples of domain-specific specifications and functionalities for an educational app.
3. Explain the significance of asking critical questions about future functions and constraints when establishing specifications for an application.
4. When establishing functionalities for user registration in an e-commerce app, what critical details should be considered? Discuss whether asking for the complete name, age, phone, or just the email is sufficient and why.
5. When designing functionalities for a quiz feature, what considerations should be made regarding retaking quizzes? Discuss the decision-making process for shuffling questions and/or answers and providing feedback or correct answers based on the possibility of retakes.

Choose a programming language corresponding to target platform (MC 3.4.B.2)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Choose a programming language corresponding to target platform. Code: MC 3.2.B.2
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.2)

Learning Outcomes (ref. Level 3-4 LOs 3.4.16 - 3.2.17):

- Is aware of criteria of choosing a programming language corresponding to target platform.
- Choose a specific programming language corresponding to target platform.

Description (MC 3.4.B.2)

At the intermediate level of programming expertise programmers should demonstrate awareness of the criteria for choosing a programming language aligned with the target platform. Recognizing that different languages excel in various application domains, developers consider factors such as whether the application is a web or mobile app, desktop software, game, or data analysis tool. Additionally, they are mindful of the strengths and weaknesses of programming languages in relation to the chosen platform, such as Java/Kotlin for Android or Swift for iOS.

Secondly, developers actively choose a specific programming language based on thorough research into application constraints, the target audience, and used platforms. They consider financial aspects, distinguishing between free and commercial options, and evaluate the frequency of updates for both programming languages and Integrated Development Environments.

Questions (MC 3.4.B.2)

1. How do the factors like strengths, weaknesses, and areas of expertise of programming languages contribute to making an informed decision based on the target platform?
2. Provide examples of programming languages that are well-suited for specific platforms.
3. When choosing a programming language corresponding to the target platform, what factors should be considered regarding application constraints, the target audience, and used platforms?
4. Describe the steps you would take to seek out information about application constraints related to the target group and used platforms.
5. When choosing a specific programming language, how do you evaluate the financial aspects, including whether it is free or commercial, and the frequency of updates?

Choose an IDE corresponding to programming language (MC 3.4.B.3)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Choose an IDE corresponding to programming language. Code: MC 3.2.B.3
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.3)

Learning Outcomes (ref. Level 3-4 LOs 3.4.18 - 3.2.19):

- Is aware of criteria of choosing an IDE.
- Choose the right IDE for a specific programming language.

Description (MC 3.4.B.3)

At the intermediate level the developers should know the diverse criteria for selecting an IDE, considering factors such as association with a programming language, platform compatibility, programming support, integrated tools, and community support. Their awareness extends to the importance of integration with external tools and consideration of IDE ecosystems. Individuals understand that personal preferences play a crucial role in the selection process, advocating for testing different IDEs to determine comfort and productivity.

Moreover, at this level, developers can choose the right IDE for a specific programming language, aligning the choice with project needs and personal/team preferences. They demonstrate familiarity with popular IDEs for various languages, showcasing a nuanced understanding of selecting the most suitable tools for diverse application projects.

Questions (MC 3.4.B.3)

1. Enumerate at least three criteria for choosing an Integrated Development Environment (IDE) when embarking on a new programming project.
2. Explain the significance of checking an IDE's compatibility with external tools or frameworks. Provide an example of a scenario where seamless integration with external tools is crucial for enhancing the development workflow.
3. Describe the concept of an IDE ecosystem and how it can benefit developers.
4. How might individual preferences impact the productivity and comfort level of a developer, and why is it recommended to test different IDEs before making a final selection?
5. Select one programming language from the list (Java, Python, JavaScript, C++, C#, Swift) and recommend the most suitable IDEs based on your understanding of their features and capabilities. Explain the rationale behind your choices, considering the language's requirements and the team's preferences.

Programming language syntax (MC 3.4.B.4)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Programming language syntax. Code: MC 3.2.B.4
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 6 – Maximum 8 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.4)

Learning Outcomes (ref. Level 3-4 LOs 3.4.20 - 3.2.22):

- Is aware of the definition of the syntax for a programming language.
- Is aware of the role of the syntax for a programming language.
- Is aware of the main rules of a language's syntax.

Description (MC 3.4.B.4)

At the intermediate level of programming expertise in digital content creation, developers possess advanced knowledge regarding the syntax of programming languages. They are aware that syntax entails a set of rules governing code structure and grammar, crucial for compiler or interpreter comprehension. Recognizing syntax's seven roles, developers understand its importance in parsing, error detection, code readability, language consistency, defining behavior, reflecting language evolution, and ensuring compatibility with tools and frameworks.

Additionally, programmers are aware of the main rules governing a language's syntax. This includes understanding aspects such as comments for explanatory notes, terminators and separators, keywords with specific meanings, identifier rules for naming variables and functions, operator rules for performing operations, statements and expressions for instructions, code blocks defined through indentation or braces, and the significance of case sensitivity. This heightened awareness enables developers to write coherent and error-free code, fostering readability, consistency, and adherence to best practices in programming applications.

Questions (MC 3.4.B.4)

1. What does the term "syntax" mean in the context of a programming language?
2. How would you describe the role of syntax in ensuring code validity and understandability by a compiler or interpreter?
3. Explain the role of syntax in parsing source code and its significance in the compilation process.
4. How does syntax contribute to error detection in programming?
5. In what ways does a well-defined syntax ensure language consistency?
6. How do language rules, encompassed by syntax, influence the behavior and semantics of a developed application?
7. Provide examples of how syntax rules align with best practices and helps avoid ambiguity in code.
8. Describe the purpose of comments in a programming language and their role in code interpretation.
9. How do terminators and separators contribute to the clarity and organization of code, especially in complex statements or lists?

Data types (MC 3.4.B.5)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Data types. Code: MC 3.2.B.5
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 4 – Maximum 6 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.5)

Learning Outcomes (ref. Level 3-4 LOs 3.4.23 - 3.2.24):

- Is aware of the classification of data types and their characteristics.
- List the main data type and their characteristics in a language program.

Description (MC 3.4.B.5)

At the intermediate level of expertise in programming, the developers should recognize that data types categorize variables or expressions based on their values, determining permissible operations, value ranges, and memory requirements. They understand the pivotal role of data types in efficient memory usage, ensuring data accuracy, and influencing overall program functionality. The programmers appreciate that the available data types in a programming language directly impact the operations applicable to data and shape the program's behavior.

Additionally, the developers demonstrate the ability to enumerate main data types, explaining their representations, and distinguishing between internal and external forms. They provide examples, such as Integer for whole numbers, Floating-Point for real numbers, Character for individual characters, String for text sequences, Boolean for binary decisions, Array for organized collections, Pointer for memory addresses, and Struct/Record for composite user-defined types. Also, they describe internal representations, such as memory storing in binary representation.

Questions (MC 3.4.B.5)

1. What is the fundamental role of data types in a programming language, and why are they considered building blocks for programming?
2. Provide a comprehensive list of main data types used in programming and outline their characteristics.
3. Choose a specific programming scenario and explain which data types would be most suitable, justifying your choices based on their characteristics.
4. Differentiate between internal and external representations of data types, using examples to illustrate your understanding.
5. In the context of developing a web application that handles user data, propose an appropriate data type for storing user ages, ensuring both efficiency and accuracy.
6. Elaborate on the roles of advanced data types like Pointers and Struct/Record in programming.
7. Provide an example scenario where the use of an array is more advantageous than other data types and justify your choice.

Variables and operators (MC 3.4.B.6)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Variables and operators. Code: MC 3.2.B.6
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 4 – Maximum 6 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.6)

Learning Outcomes (ref. Level 3-4 LOs 3.4.23 - 3.2.24):

- Is aware on how to declare a variable.
- Is aware on how to use operators.

Description (MC 3.4.B.6)

At the intermediate level, the developers should possesses the understanding that declaring variables is a fundamental step in programming, ensuring they are defined before usage. They recognize an identifier as a naming mechanism for various programming entities, adhering to defined rules such as character set allowances (letters, digits, underscores) and length considerations. Also, they demonstrate awareness of variable naming conventions, essential for maintaining code clarity and consistency across programming languages.

The programmers should know that operators are symbolic representations or keywords facilitating specific operations on operands, which could be values or variables. They should be familiar with the common categories of operators, encompassing arithmetic, comparison, logical, assignment, bitwise, conditional, cast, instanceof, member access, and address-of. They understand the types of operators, including unary, binary, and ternary, recognizing their distinct roles in programming logic. Also, they understand the concept of operator notation and its variability (infix, pre-fixed, post-fixed), recognizing that the notation style is language dependent.

Questions (MC 3.4.B.6)

1. Explain the significance of declaring variables in programming.
2. Outline the rules for defining a variable name and provide an example of a valid variable name considering the allowed characters.
3. Define what an identifier is in the context of programming.
4. Describe the role of operators in a programming language.
5. Enumerate the common categories of operators and furnish examples for each category, illustrating their distinct functionalities.
6. Differentiate between unary, binary, and ternary operators.
7. Explain the concept of operator notation and its variations (infix, pre-fixed, post-fixed).

Control flow statements (MC 3.4.B.7)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Control flow statements. Code: MC 3.2.B.7
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 4 – Maximum 6 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.7)

Learning Outcomes (ref. Level 3-4 LOs 3.4.27):

- List and explain the control flow statements.

Description (MC 3.4.B.7)

At the intermediate level of expertise in programming, people should understand that control flow statements are crucial for designing programs that execute different actions based on specific conditions. An intermediate-level programmer possesses the knowledge to employ various control flow statements that shape the execution sequence of a program. This includes to comprehend the significance of control flow statements, which are integral constructs dictating the program's execution order. These statements provide the means to implement conditional branching and looping behavior. The developers should be proficient in the various types of control flow statements, including conditional statements (if, if...else), looping statements (do...while, while, for), loop control statements (break, continue), control statements (goto), and exception handling statements (try...catch).

They should be capable of understand and apply the logic behind each control flow statement. This involves following the conditions under which branching occurs, the iteration criteria for loops, and the role of control statements in altering the program's flow.

Intermediate programmers should demonstrate the ability to make informed decisions on when to use specific control flow statements. They understand the implications of their choices on program behavior, readability, and maintenance.

Questions (MC 3.4.B.7)

1. Explain the significance of control flow statements in programming and how they contribute to the overall structure and logic of a program.
2. Provide an example scenario where effective use of control flow enhances program functionality.
3. List and briefly explain each type of control flow statement, including conditional, looping, loop control, control, and exception handling.
4. Illustrate situations where each type is appropriately used.
5. Provide a real-world analogy or example to illustrate using a control flow statement.
6. For looping statements (do...while, while, for), describe the fundamental differences and scenarios where each is most suitable.
7. Explain the purpose and operation of exception handling statements (try...catch) in controlling program flow during error situations.

Run source file (MC 3.4.B.8)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Run source file. Code: MC 3.2.B.8
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 2 – Maximum 4 hrs
Level of the learning experience leading to the micro-credential	INTERMEDIATE
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.B.8)

Learning Outcomes (ref. Level 3-4 LOs 3.4.28 - 3.2.30):

- Is aware on how to write a simple source file.
- Is aware on how to run a source file.

Description (MC 3.4.B.8)

At the intermediate level of expertise, developers should be aware of creating a new source file with the correct file extension for the chosen programming language. They can apply syntax rules to craft a simple application that may involve variables, functions, control flow statement. Proficient in text editors or integrated development environments (IDEs), they follow a systematic process: creating a new file, writing code, saving it with a meaningful name and the appropriate extension, and selecting a location for easy retrieval. The developer is able to organize the code into projects and manage project files. This includes creating project structures, adding dependencies, and configuring build paths.

Skills extends to understanding the nuanced steps required based on the programming language, tools, and development environment in use. Proficient in compiling/interpreting, if necessary, to generate an executable file. When running the program, individuals can navigate both command-line interfaces and IDEs. In the command-line interface, they use the appropriate command, while in the IDE's interface, they leverage the suitable option to execute the application.

Questions (MC 3.4.B.8)

1. Can you describe the essential steps involved in creating a new source file for a programming language?
2. How do you decide on a meaningful name for your source file, and what are your considerations when choosing the correct file extension?
3. How do you decide on a suitable location on your computer to save the file for easy retrieval later?
4. Can you outline the general steps to run a source file, taking into consideration the programming language, installed tools, and the development environment you are using?
5. Explain the concept of compilation in programming, and under what circumstances would you need to compile your source code before running it?
6. Describe the process of running a program in a command-line interface (terminal)/Integrated Development Environment (IDE).
7. Describe the steps you would take to organize code into projects, including creating a project structure, adding dependencies, and configuring build paths.

ADVANCED LEVEL

(Level 5 and Level 6)

Methods/functions (MC 3.4.C.1)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Methods/functions. Code: MC 3.2.C.1
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 4 – Maximum 6 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.1)

Learning Outcomes (ref. Level 5-6 LOs 3.4.31 - 3.2.33):

- Declare methods/functions.
- Define methods/functions.
- Apply methods/functions.

Description (MC 3.4.C.1)

At the advanced level of expertise in programming, the developers should understand that method declaration involves recognizing key elements such as the method name, return type, parameters, and method body. This knowledge spans various programming languages, ensuring consistency in method declaration practices.

Proficiency extends to method definition, where expertise lies in crafting the method's behavior, logic, and functionality based on the earlier declaration. This encompasses creating a method body, formulating algorithms and logic for task execution, and, if applicable, including a return statement to specify the value sent back to the caller.

Application proficiency is demonstrated through the skill of invoking methods in different parts of the app, executing designated actions. This involves a comprehensive understanding of how methods interact within the codebase, contributing to the overall functionality and efficiency of the program.

Questions (MC 3.4.C.1)

1. What are the essential elements of a method declaration in programming?
2. Explain the components of a method definition in programming, including the significance of the method body, algorithm, and the return statement.
3. Provide an example scenario where calling a method is crucial for executing specific actions within a program. Describe the steps involved in calling a method from different parts of the codebase.
4. Across various programming languages, what are the common key elements consistently found in method declarations?
5. Demonstrate your understanding of the interplay between method declaration and definition.

Input/output (I/O) operations (MC 3.4.C.2)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Input/output (I/O) operations. Code: MC 3.2.C.2
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 4 – Maximum 6 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.2)

Learning Outcomes (ref. Level 5-6 LOs 3.4.34 - 3.2.35):

- Is aware of input/output (I/O) operations.
- Perform input/output (I/O) operations.

Description (MC 3.4.C.2)

At the advanced level of expertise in programming, the developers should know that input/output (I/O) operations in programming are the means of reading data from external sources or sending data to external destinations, such as interacting with users, files, network resources, databases, and other systems. Recognizes the fundamental role I/O operations play in data processing. Advanced I/O operations proficiency demonstrate the ability to perform various I/O operations, including reading input from diverse sources (keyboard, files, and network connections), writing output (to the console, files, network, and databases), implementing formatted output for human-readable displays, applying error handling techniques for managing exceptions during I/O operations, conducting serialization and deserialization processes for data storage or transmission, understanding and utilizing standard streams (input - stdin, output - stdout, and error - stderr).

Questions (MC 3.4.C.2)

1. What is the purpose of input/output (I/O) operations in programming, and why are they crucial for software applications interacting with external systems?
2. Demonstrate the steps involved in performing I/O operations. Provide examples of reading input from different sources (keyboard, files, network) and writing output to various destinations (console, files, network, databases).
3. How would you display data in a human-readable format, applying formatting to numbers and dates during output operations?
4. Define serialization and deserialization in the context of programming.
5. How are input (stdin), output (stdout), and error (stderr) streams utilized in programming?

Libraries and modules (MC 3.4.C.3)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Libraries and modules. Code: MC 3.2.C.3
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.3)

Learning Outcomes (ref. Level 5-6 LOs 3.4.36 - 3.2.37):

- Is aware of libraries and modules.
- Operate with libraries and modules.

Description (MC 3.4.C.3)

At the advanced level of expertise in programming, the developers should be able to define the concept of libraries and modules in programming. Explain the role of libraries as collections of pre-compiled code with specific functionalities and modules as self-contained units of code. They also should be able to discuss the importance of factors like compatibility, licensing, documentation, and community support when considering the use of libraries and modules.

Secondly, they should explain the impact of operating with libraries and modules on the speed and efficiency of software development. Discuss how utilizing pre-written code allows developers to focus on core logic and implement unique aspects of an application more rapidly.

Questions (MC 3.4.C.3)

1. Explain the concept of libraries and modules in programming. What distinguishes a library from a module, and how are they utilized to enhance software applications?
2. Discuss the significance of considering factors like compatibility, licensing, documentation, and community support when working with libraries and modules.
3. Describe the benefits of reusing code through libraries and modules.
4. How can operating with libraries and modules contribute to faster development?
5. How can a programmer actively participate in and benefit from community efforts to develop and improve libraries?

Programming paradigms (MC 3.4.C.4)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Programming paradigms. Code: MC 3.2.C.4
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 50%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.4)

Learning Outcomes (ref. Level 5-6 LOs 3.4.38):

- Enumerate and explain the programming paradigms.

Description (MC 3.4.C.4)

At the advanced level of expertise in programming, the developers should know that programming paradigms are fundamental approaches to structuring and organizing code, representing different philosophies and methodologies for designing and implementing software. Each paradigm provides a distinct way of thinking about program design, emphasizing particular principles and practices.

Programmers should enumerate and explain the characteristics of main programming paradigms, including imperative, object-oriented, functional, logical, concurrent, parallel, procedural, event-driven, and structured programming. Highlight key concepts and examples associated with each paradigm. Also, they should be able to highlight the fact that many modern programming languages and frameworks support a combination of programming paradigms.

Questions (MC 3.4.C.4)

1. Enumerate the main programming paradigms and briefly explain the characteristics of each. Provide an example language for each paradigm.
2. Describe the key features of imperative programming. Provide examples of programming languages that follow this paradigm.
3. Explain the core principles of functional programming. Provide examples of functional programming languages.
4. Discuss the characteristics of Object-Oriented Programming (OOP). Provide examples of OOP languages.
5. Exemplify a modern programming languages or frameworks that support a combination of programming paradigms.
6. Provide an example scenario where a combination of paradigms might be advantageous.

Imperative and Object-Oriented Programming (MC 3.4.C.5)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Imperative and Object-Oriented Programming. Code: MC 3.2.C.5
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.5)

Learning Outcomes (ref. Level 5-6 LOs 3.4.39 - 3.2.40):

- Explain the concepts for imperative procedural programming.
- Explain the concepts for Object-Oriented Programming (OOP).

Description (MC 3.4.C.5)

At the advanced level of expertise in programming, the developers should know that imperative procedural programming, a fundamental paradigm, focuses on sequencing steps and procedures to accomplish tasks. Key concepts encompass sequencing, variables, control structures, structured programming, modularity, and procedures/functions. A robust understanding of these concepts enables advanced programmers to create well-organized, modular, and extensible code that efficiently addresses complex problems.

Also, the programmers should know that Object-Oriented Programming (OOP) is a paradigm emphasizing objects, classes, and methods for code organization and problem-solving. Key concepts encompass objects and classes, encapsulation, inheritance, polymorphism, method overriding, method overloading, constructors and destructors, message passing, visibility modifiers.

Questions (MC 3.4.C.5)

1. Explain the concept of "sequencing" in imperative procedural programming.
2. How does "modularity" contribute to the effectiveness of imperative procedural code?
3. Discuss the significance of "structured programming" in improving code maintainability and readability.
4. Elaborate on the concept of "encapsulation" in OOP and provide an example illustrating how it enhances code security and organization.
5. Discuss the principles of "inheritance" in OOP and explain how it facilitates code reuse and promotes a hierarchical structure.
6. Describe the role of "constructors and destructors" in OOP, emphasizing when and why they are essential in class design.

Logic and Functional Programming (MC 3.4.C.6)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Logic and Functional Programming. Code: MC 3.2.C.6
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.6)

Learning Outcomes (ref. Level 5-6 LOs 3.4.41 - 3.2.42):

- Explain the concepts for declarative logic.
- Explain the concepts for declarative functional.

Description (MC 3.4.C.6)

At the advanced level of expertise in programming applications, developers should possess a deep understanding of declarative logic programming emphasizing on expressing relationships and facts without explicitly specifying control flow. This paradigm, often exemplified by languages like Prolog, allows developers to declare the logic and let the system determine the execution sequence. The programmer is adept at constructing logic-based systems, utilizing facts, and defining relationships without the need for explicit procedural instructions. This knowledge is crucial for tasks involving rule-based systems, where expressing the logic of relationships takes precedence over specifying step-by-step execution.

At declarative functional programming, the advanced programmer should understand it center on expressing computation as mathematical functions without relying on mutable state or side effects. The functional paradigm, exemplified by languages like Haskell and Lisp, promotes a more concise and abstract representation of algorithms. The programmer is adept at creating robust, modular, and scalable systems by leveraging the power of higher-order functions, immutability, and a focus on pure computations.

Questions (MC 3.4.C.6)

1. Can you articulate the key principle of declarative logic programming and how it differs from imperative paradigms, emphasizing the expression of relationships and facts?
2. Provide an example scenario where declarative logic programming, such as Prolog, would be more advantageous than an imperative approach.
3. Describe the fundamental principles of declarative functional programming.
4. How does functional programming paradigm enhance code clarity and maintainability compared to imperative or object-oriented approaches?

Concurrent programming (MC 3.4.C.7)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Concurrent programming. Code: MC 3.2.C.7
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 3 – Maximum 5 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.7)

Learning Outcomes (ref. Level 5-6 LOs 3.4.43 - 3.2.44):

- List the techniques for concurrency.
- Carry out concurrency.

Description (MC 3.4.C.7)

Advanced expertise in programming applications involves a comprehensive understanding of concurrency techniques and the ability to implement them effectively. The developers should understand techniques of multi-threading (threads operate within a single process, sharing memory space) and multi-processing (use of separate processes with distinct memory spaces for concurrency). They should identify independent tasks and assesses which parts of the program can run concurrently without conflicts. Then they should determine shared resources and applies synchronization mechanisms (locks, semaphores) to manage access and prevent data races and ensure thread-safe data access, employing safety and synchronization measures.

Questions (MC 3.4.C.7)

1. Describe the fundamental concept of concurrency and its role in program efficiency.
2. Differentiate between multi-threading and multi-processing, explaining how each technique achieves concurrency.
3. Choose between multi-threading and multi-processing based on a given application's requirements and constraints.
4. Given a code snippet, identify potential independent tasks suitable for concurrent execution.
5. Evaluate a piece of code and identify shared resources that may lead to data races. Propose synchronization mechanisms to mitigate these issues.
6. Analyze a code snippet and identify potential areas where deadlocks or race conditions may occur. Propose solutions to prevent or resolve these issues.

Testing and debugging (MC 3.4.C.8)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Testing and debugging. Code: MC 3.2.C.8
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 8 – Maximum 10 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.8)

Learning Outcomes (ref. Level 5-6 LOs 3.4.45 - 3.2.48):

- Testing and debugging.
- Debug an application.
- Test an application.

Description (MC 3.4.C.8)

The advanced-level programmers possess a comprehensive understanding and application of testing and debugging phases. They recognize testing as a systematic process to identify defects, ensuring software adheres to specifications and behaves as intended. Also, thy understand debugging as the targeted resolution of defects, involving code examination, execution tracing, and issue root cause identification. Knowing and applying debugging techniques like, use of print statements strategically to output variable values, interactive debugging in IDEs or tools, utilizing breakpoints and variable inspection, implements logging with libraries to record program execution details for in-depth analysis, code review collaboration.

Secondly, at advanced level, the developers demonstrate expertise in testing methodologies like, unit testing (automated through frameworks), integration testing (to validate interactions between diverse units and modules), functional testing (scenarios), regression testing (to prevent the introduction of defects with new changes), performance testing (to evaluate software responsiveness under varying conditions), security testing (identifying vulnerabilities and mitigating potential risks).

Questions (MC 3.4.C.8)

1. How would you articulate the role of testing and debugging in the software development lifecycle?
2. Describe the debugging techniques you would employ to identify and resolve issues in a complex codebase.
3. When faced with a bug in an application, how would you leverage print statements, interactive debugging in IDEs, logging libraries, and code reviews to efficiently identify and address the root cause of the issue?
4. Explain the importance of various testing types (unit testing, integration testing, functional testing, regression testing, performance testing, and security testing) in ensuring a robust and reliable software application. How do these types complement each other?
5. How do you approach testing and debugging processes, considering their iterative nature?

Compiled vs. interpreted programming languages (MC 3.4.C.9)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Compiled vs. interpreted programming languages. Code: MC 3.2.C.9
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 6 – Maximum 8 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.9)

Learning Outcomes (ref. Level 5-6 LOs 3.4.49 - 3.2.52):

- Explain the differences between compiled and interpreted (scripting).
- Characteristics of compiled language.
- Characteristics of interpreted language.
- Is aware of typed of the programming language.

Description (MC 3.4.C.9)

Advanced expertise in programming applications involve distinguish between compiled and interpreted languages. Compiled languages undergo a pre-execution compilation step, translating source code into efficient machine code or bytecode. Interpreted languages, conversely, execute code in real-time, line-by-line, offering rapid development cycles and ease of debugging.

Advanced professionals recognize the characteristics of compiled languages, emphasizing superior performance, permits multiple executions without recompilation, but reduced portability due to platform-specific compilation. Examples include C, C++, Rust, and Go. Interpreted languages prioritize ease of development. They exhibit generally slower performance, but modern interpreters incorporate optimizations. Notably, they boast enhanced portability as the same code can run on various platforms. Examples encompass Python, Ruby, JavaScript, and PHP.

Moreover, advanced practitioners comprehend the distinction between statically and dynamically typed languages. Statically typed languages enforce type declarations at compile-time, ensuring type correctness, exemplified by Java or C++. Dynamically typed languages, typified by Python or JavaScript, determine types at runtime, offering flexibility for variable types to change dynamically during execution.

Questions (MC 3.4.C.9)

1. Describe the key differences between compiled and interpreted languages in the context of the pre-execution compilation step.
2. Considering the characteristics of compiled languages, discuss the trade-off between superior performance and reduced portability.
3. Explain the advantages of interpreted languages, emphasizing their real-time execution and rapid development cycles.
4. Discuss the implications of statically typed languages, emphasizing the role of compile-time type checking.
5. How does dynamic typing accommodate changes in variable types during program execution, and what advantages does this flexibility bring to certain development scenarios?

Markup languages (MC 3.4.C.10)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Markup languages. Code: MC 3.2.C.10
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 6 – Maximum 8 hrs
Level of the learning experience leading to the micro-credential	ADVANCED
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.C.10)

Learning Outcomes (ref. Level 5-6 LOs 3.4.53 - 3.2.54):

- Explain the concepts for markup languages.
- Transformation and styling of markup languages.

Description (MC 3.4.C.10)

Advanced proficiency in programming applications involves a nuanced understanding of markup languages. These languages, such as HTML, XML, Markdown, and LaTeX, serve to annotate text, defining its structure, presentation, and behavior within a document. Mastery in markup encompasses comprehension of essential concepts, including markup tags, attributes, nesting, validation, and ensuring interoperability across diverse platforms and devices.

Furthermore, an advanced professional demonstrates the ability to transform and style content using technologies like Cascading Style Sheets (CSS) for web documents and Extensible Stylesheet Language Transformations (XSLT) for XML documents. This skill involves leveraging transformation technologies to enhance the visual presentation and structural representation of content. Proficient application of these techniques ensures the creation of accessible, well-organized, and aesthetically pleasing digital content across various contexts and platforms.

Questions (MC 3.4.C.10)

1. Can articulate the fundamental concepts of markup languages?
2. Demonstrates understanding of the role of markup languages in defining the structure, presentation, and behavior of text within a document.
3. Can explain how markup languages contribute to interoperability across diverse platforms and devices.
4. Exhibits proficiency in using Cascading Style Sheets (CSS) for styling web documents.
5. Demonstrates competence in employing Extensible Stylesheet Language Transformations (XSLT) for transforming and styling XML documents.
6. Shows the ability to apply transformation technologies effectively to enhance the visual presentation and structural representation of content.

EXPERT LEVEL

(Level 7 and Level 8)

Design solution for complex problem (MC 3.4.D.1)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Design solution for complex problem. Code: MC **3.2.D.1**
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 8 – Maximum 10 hrs
Level of the learning experience leading to the micro-credential	EXPERT
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.D.1)

Learning Outcomes (ref. Level 7-8 LOs 3.4.5% - 3.2.57):

- Establish statements and requirements for complex problems with limited definition.
- Design solutions to solve complex problems with many interacting factors.
- Designs the implementation of complex computer systems.

Description (MC 3.4.D.1)

Highly specialized professionals are able to articulate critical questions to unveil complexities in poorly defined or evolving problem statements in specialized domains. Demonstrates the ability to iteratively refine problem statements by collaborating with domain experts and analyzing user needs. Integrates systematic and strategic approaches to design solutions for complex problems with numerous interacting factors. Shows proficiency in problem decomposition, data analysis, mathematical modeling, and collaboration across disciplines. Applies machine learning and artificial intelligence techniques for analyzing and predicting complex interactions.

Designs and integrates hardware, software, and networking components to create functional and secure computer systems. Demonstrates responsibility in meeting size, performance, and complexity requirements, ensuring scalability and ease of maintenance. Evaluates and recommends new technologies, considering factors such as security features, long-term support, integration, and budget constraints. Makes informed decisions on selecting target platforms, languages, and frameworks based on ecosystem considerations and project requirements.

Manages personal resources effectively, considering the availability of libraries, security features, long-term support, integration, and budgetary constraints. Maintains open and transparent communication with stakeholders, ensuring responsible development practices.

Questions (MC 3.4.D.1)

1. How do you approach iterative refinement of problem statements, especially when collaborating with domain experts and considering user needs?
2. Describe a situation where you successfully applied a systematic and strategic approach to design solutions for a complex problem with multiple interacting factors.
3. How do you ensure proficiency in problem decomposition, data analysis, mathematical modeling, and collaboration across diverse disciplines in your solution design process?
4. Provide examples of how you've applied machine learning and artificial intelligence techniques to analyze and predict complex interactions in your projects.
5. How do you ensure that the computer system you design meets size, performance, and complexity requirements, ensuring scalability and ease of maintenance?
6. How do you make informed decisions when selecting target platforms, languages, and frameworks, considering ecosystem considerations and project requirements?

Project management (MC 3.4.D.2)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Project management. Code: MC 3.2.D.2
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 8 – Maximum 10 hrs
Level of the learning experience leading to the micro-credential	EXPERT
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.D.2)

Learning Outcomes (ref. Level 7-8 LOs 3.4.58 - 3.2.60):

- Supervises the implementation of complex computer systems.
- Supervise compliance with specifications.
- Implement project management methods.

Description (MC 3.4.D.2)

Highly specialized expertise in programming applications involves overseeing the implementation of complex computer systems with a keen awareness of diverse languages and technologies. The manager emphasizes compliance with specifications, ensuring precise adherence to coding best practices, project structures, version control, and other concepts vital for creating sophisticated, maintainable applications.

Implementation of project management methods encompass tasks such as comprehensive project planning, defining goals, and developing timelines. Team management involves assembling skilled teams, assigning roles, and fostering collaboration. The project manager engages in risk management, quality assurance, and ongoing monitoring and control, ensuring that the project stays on track. Resource management and meticulous documentation contribute to successful software project execution, demonstrating a holistic approach to the development life cycle.

Questions (MC 3.4.D.2)

1. How you ensured compliance with specifications, coding best practices, and project structures in your previous projects.
2. How did this commitment contribute to the creation of sophisticated and maintainable applications?
3. Explain your approach to comprehensive project planning, including how you define project goals, develop timelines, and allocate resources.
4. How has your project management proficiency contributed to successful software project execution?
5. Describe your experience in assembling skilled teams, assigning roles, and fostering collaboration within a software development project.
6. How you handle risk management and quality assurance in software development projects?

Leadership. Creativity (MC 3.4.D.3)

Basic Information

Identification of the learner	Any Citizen
Title and code of the micro-credential	Leadership. Creativity. Code: MC 3.2.D.3
Country(ies)/Region(s) of the issuer	IRELAND, ITALY, CYPRUS, GREECE, ROMANIA http://dsw.projectsgallery.eu
Awarding body(ies)	DSW Consortium Project Number: 101087628
Date of issuing	Nov 2023
Notional workload needed to achieve the learning outcomes	Minimum 8 – Maximum 10 hrs
Level of the learning experience leading to the micro-credential	EXPERT
Type of assessment	Automatically marked Questions. Number of Questions: 10 Passing Score: 75%
Form of participation in the learning activity	Online Asynchronous
Type of quality assurance used to underpin the micro-credential	Peer Review

Learning outcomes (MC 3.4.D.3)

Learning Outcomes (ref. Level 7-8 LOs 3.4.61 - 3.2.62):

- Guide others in the analysis and development of applications.
- Propose new ideas and processes to the field.

Description (MC 3.4.D.3)

At the highly specialized level of expertise in programming applications, the professional takes on a leadership role by guiding others in the analysis and development of applications. They actively contribute to professional practices, fostering continuous learning and knowledge updates within the team. Through mentorship, collaborative projects, and the creation of documentation and best practices, they empower their organization to excel. This involves developing reusable patterns, contributing to open-source projects, and advocating for ethical and compliant practices, automation, and continuous integration.

Additionally, the expert encourages innovation by proposing new ideas and processes to the field. They emphasize the importance of creativity, problem-solving, research, and cross-disciplinary collaboration. Their focus extends to continuous improvement, efficiency, cost reduction, scalability, user-centered design, sustainability, and risk management. Overall, this professional strives to bring positive changes, advancements, and efficiencies, ensuring their organization remains competitive and adaptable in a rapidly changing environment.

Questions (MC 3.4.D.3)

1. How does the professional actively contribute to continuous learning and knowledge updates within the team, and what strategies do they employ to foster a culture of learning and development?
2. Can you provide examples of collaborative projects and the impact of your contributions on the team's success and project outcomes?
3. In what ways does the individual empower their organization to excel through the development of reusable patterns and contributions to open-source projects?
4. How does the expert advocate for ethical and compliant practices, automation, and continuous integration in the development process?
5. Can you share instances where you proposed new ideas and processes to the field, emphasizing creativity, problem-solving, and cross-disciplinary collaboration?

DSW
DIGITAL SKILLS WALLET

Co-funded by
the European Union

APPENDIX I LEARNING OUTCOMES FOR COMPETENCE AREA: DIGITAL CONTENT CREATION

COMPETENCE: 3.4 Programming

INTRODUCTION:

Programming languages encompass a wide range of topics and features that enable developers to communicate instructions to computers.

Different programming languages may emphasize certain topics more than others. Learning multiple languages can give you a broader perspective on programming and help you choose the best tool for different projects.

Software applications are developed for a specific application domain (e.g., finance, medicine, automotive, Internet, security). Knowledge of that field is important to understand and formalize and develop the functionalities of the future application.

Developing the knowledge areas and attitudes for developing applications into a programming language through formal education, training programs, and practical experience can enhance an individual's programming skills, but to perform in any programming language is needed a lot of practice and experience.

PREREQUISITES

To develop knowledge, skills and attitudes related to the competency COPYRIGHT AND LICENCES several areas serve well as prerequisites. These include:

1. **Basic Computer Skills**: You should be familiar with how to use a computer, navigate the file system, create and manage files and folders, and perform basic tasks like copying, pasting, and installing software.
2. **Mathematics**: While not all programming languages require advanced mathematics, having a basic understanding of arithmetic, algebra, and logic can be beneficial for solving problems and understanding algorithms.
3. **Logical Thinking**: Programming involves breaking down problems into smaller logical steps and devising solutions. Developing strong logical thinking skills will help you write more efficient and organized code.
4. **Problem-Solving Skills**: Programming is all about problem-solving. Being able to approach problems methodically and think critically will be essential in writing effective code.
5. **English Proficiency**: Most programming resources, tutorials, and documentation are available in English. A good grasp of the English language will make it easier to access learning materials and communicate with the programming community.
6. **Basic Understanding of Computers and Algorithms**: Familiarity with how computers process data and execute instructions will provide a foundation for understanding how programming languages interact with hardware. Understanding basic algorithms will also be helpful in programming.
7. **Operating systems**: like Windows, macOS, or Linux. An operating system (OS) is a fundamental software component that manages computer hardware and provides services and interfaces for user programs. It acts as an intermediary between applications and the hardware, allowing users to interact with the computer's resources in a more user-friendly and efficient manner.

COMPETENCE AREA: 3. DIGITAL CONTENT CREATION		DIMENSION 3: PROFICIENCY LEVEL
COMPETENCE: 3.4 PROGRAMMING		
1	At basic level and with guidance, I can:	• identify the main types of application that can be developed into a programming language, • enumerate the main programming languages, • identify what is an IDE – Integrated Development Environment.
2	At basic level and with autonomy and appropriate guidance where needed, I can:	• identify the main characteristic of different types of applications, • detect the basic features and functionalities for different IDEs, • identify the application domain as real life area on which the application is intended to apply, • explain what language programming documentation is.
3	On my own and solving straightforward problems, I can:	• explain the main specifications/functionalities for an application into a particular domain, • choose a specific programming language corresponding to platform, • choose the right IDE for a specific programming language, • explain the syntax of a specific programming language.
4	Independently, according to my own needs, and solving well-defined and non-routine problems, I can:	• explain the data types, • explain and declare the variables, • explain how to use operators, • apply control flow statements.
5	As well as guiding others, I can:	• define, declare, and apply methods/functions, • perform input/output (I/O) operations, • carry out libraries and modules, • enumerate and explain the programming paradigms.
6	At advanced level, according to my own needs and those of others, and in complex contexts, I can:	• explain the concepts for imperative Object-Oriented Programming (OOP) or imperative procedural or declarative logic or declarative functional, • carry out concurrency, • testing and debugging, • explain the differences between compiled and interpreted (scripting), • explain the concepts for markup languages.
7	At highly specialized level, I	• designs and supervises the implementation of complex computer systems,

	can:	• create applications to solve complex problems with limited definition that are related to an application domain, • integrate my knowledge to contribute to professional practices and knowledge and to guide others in the analysis and development of applications.
8	At the most advanced and specialised level, I can:	• create solutions to solve complex problems with many interacting factors that are related to an application domain, • propose new ideas and processes to the field.

COMPETENCE AREA: 3. digital content creation			
COMPETENCE: 3.4 PROGRAMMING			
FOUNDATION			
Learning Outcome	Level	K – S - A	Example
1. Understanding types of application that can be developed into a programming language.	L1 – L2	K	Enumerate the main types of applications regarding the implementation platform that can be developed for desktop, mobile, web, embedded. Describe what are the main differences between them (how can be used to develop applications for various systems and environments).
2. Explain the characteristics of different types of applications regarding working platform and environment.	L2	K	Enumerate applications characteristics based on their working platform (desktop, mobile, web, embedded), and environment (standalone software, optimized for touch interfaces, accessed through the internet, targeted for specific functions in devices requiring efficient resource utilization and real-time processing).
3. Is aware on how the programming language can be used to develop applications for various systems and environments.	L1 – L2	K	Recognize the difference between platform-specific and cross-platform programming languages. Some programming languages are specifically designed for certain platforms.
4. Enumerate a list with common language programming.	L1	K	Knows different example of programming languages: C, C++, C#, Java, Kotlin, JavaScript, HTML, XML, React, Python, Ruby, Prolog, Lisp, Scala.

5. Identify what is an IDE – Integrated Development Environment	L1	K	Knows that IDE stands for "Integrated Development Environment." It is a software application that provides comprehensive tools and features to facilitate the development of software applications.
6. Is aware of the main features and functionalities for different IDEs	L1 – L2	K	An Integrated Development Environment (IDE) typically offers a comprehensive set of features and functionalities to facilitate software development.
7. Find and choose an IDE corresponding with a programming language	L2	S	Knows how to find on the Internet a list with available IDEs for a specific programming language. Based on IDEs characteristics can choose an IDE.
8. Install an IDE	L2	S	Knows how to install an IDE using the predefined or custom settings.
9. Explain and find language programming documentation	L1 – L2	K	Knows that there are 4 types of language programming documentation: official language documentation, tutorials, posts/discussions, reference materials.
10. Is able to run a simple program	L2	S	Can create, edit, and save source code files with the specific extension within the IDE. Is able to build and compile the code (usually taken from official documentation) within the IDE.
11. Explain the version control	L2	S	Version control is used to keep track of code changes. Knows how to revert to a known working state if necessary.
12. Classify the application domains	L2	K	The application domains are real-life area on which the application is intended to apply.
13. Identify specific demands and constraints for a specific application domain	L2	S	Can put questions about considerations such as user requirements, data structures, algorithms, security measures, and compliance standards relevant to the application domain.

COMPETENCE AREA: 3. DIGITAL CONTENT CREATION			
COMPETENCE: 3.4 PROGRAMMING			
INTERMEDIATE			
Learning Outcome	Level	K – S - A	Example
14. Explain the main specifications/functionalities for an application into a particular domain.	L3	K	Knows that the specifications or functionalities for an application in a particular domain refer to the specific requirements and capabilities the application should have to effectively address the needs and challenges of that domain.
15. Establish the main specifications/functionalities for an application into a particular domain.	L3	A	Asks critical questions about future functions, and about constraints of application.
16. Is aware of criteria of choosing a programming language corresponding to target platform	L3	K	Aware of choosing the right programming language for your application depends on several factors (single/cross-platforms, web/mobile/desktop/).
17. Choose a specific programming language corresponding to target platform	L3	A	Seeks out information about application constrains regarding target group, used platforms. Consider the financial aspect like free/commercial, open source, frequency of updates.
18. Is aware of criteria of choosing an IDE	L3-L4	K	Knows different criteria for choosing an IDE, like association with a programming language, platform compatibility (single or cross-platform), programming support, integrated tools and plugins, UX interface,

			documentation and community, performance, license and costs.
19. Choose the right IDE for a specific programming language	L3-L4	S	Can choose the IDE depending on the project and personal/team preferences. The choice should be aligned with workflow and requirements, considering that there is no one-size-fits-all solution.
20. Is aware of the definition of the syntax for a programming language	L3	K	Knows that syntax in the context of a programming language refers to the set of rules and conventions that dictate how the code should be written to be considered valid and understandable by the compiler or interpreter.
21. Is aware of the role of the syntax for a programming language	L3-L4	K	Knows that syntax has 7 roles. 1. Parsing the source code. 2. Error detection. 3. Code readability. 4. Language consistency. 5. Language rules. 6. Language evolution. 7. Compatibility of various tools, libraries, and framework.
22. Is aware of the main rules of a language's syntax	L3-L4	K	Knows that the main rules of a programming language's syntax refer to: 1. Comments. 2. Terminators and separators. 3. Keywords. 4. Identifiers rules. 5. Operators rules. 6. Statements and expressions. 7. Code blocks. 8. Case sensitivity.

			9. Whitespace rules.
23. Is aware of the classification of data types and their characteristics	L4	K	Knows that a data type in a programming language is a classification that specifies the type of value that a variable or expression can hold. It defines the set of operations that can be performed on the data, the range of values it can represent, and the memory space required to store it.
24. List the main data type and their characteristics in a language program	L4	S	Is able to list the main data types, what they represent and what are the internal and external representation.
25. Is aware on how to declare a variable	L4	K	Knows that depending on language programming variable must be declared before to be used. Knows that an identifier is a name given to a package, class, interface, method, or variable. Knows the rules of defining a variable name.
26. Is aware on how to use operators	L4	K	Knows that operators in a programming language are symbols or keywords that represent specific operations to be performed on one or more operands. Knows the common categories of operators. Knows the types of operators: unary, binary, ternary. Knows the notation of operators (infix, pre-fixed, post-fixed).
27. List and explain the control flow statements	L4	K	Knows that control flow statements are programming constructs that determine the order in which statements are executed in a program. Know the logic of each control statement.
28. Is aware on how to write a simple source file.	L4	S	Is able to create a new source file with the file extension corresponding to language programming. Knows how to apply the syntax rule to create a simple application.
29. Is aware on how to run a	L4	S	Knows how to differentiate the exact steps depending on the used programming language, the installed tools, and the development

source file.			environment. Is able to compile to generate an executable file from source code. Is able to run the program in (a) command-line interface (terminal), (b) IDE's interface.
30. Implement project management methods	L4	S	Is able to organize the code into projects and manage project files. This includes creating project structures, adding dependencies, and configuring build paths.

COMPETENCE AREA: 3. DIGITAL CONTENT CREATION			
COMPETENCE: 3.4 PROGRAMMING			
ADVANCED			
Learning Outcome	Level	K – S - A	Example
31. Declare methods/functions	L5	K	Knows that a method declaration in programming refers to the process of defining the characteristics and behavior of a function within a class or module. Knows what the key elements of a method declaration (consistent across most programming languages) are: name, return type, parameters, method body.
32. Define methods/functions	L5	K	Knows that a method definition in programming refers to the implementation of the method's behavior, logic, and functionality that was declared earlier. A method definition includes method body, algorithm and logic, return statement.
33. Apply methods/functions	L5	S	Knows how to call a method from various parts of the code to execute the specified actions.
34. Is aware of input/output (I/O) operations	L5	K	Knows that input/output (I/O) operations in programming refers to the process of reading data from external sources (input) or sending data to external destinations (output). I/O operations are essential for interacting with users, files, network resources, databases, and other external systems.
35. Perform input/output (I/O) operations	L5-6	S	Knows how to: - read input/write output: from/to console, files, network. - format output. - handle error.

			- serialize and deserialize.
36. Is aware of libraries and modules	L5-6		Knows that refers to the process of using pre-written code components to extend the functionality of a software applications. Libraries and modules are collections of functions, classes, and resources that have been created by other developers to provide specific functionality.
37. Operate with libraries and modules	L5-6	S	Knows how to reuse the code that has already been written and tested, reducing duplication of effort. Knows how to develop faster by leveraging existing libraries and modules. Knows how to focus on core logic by using existing solutions to implement the unique aspects of an application. Knows how to search for community collaboration that develop, maintain, and update libraries.
38. Enumerate and explain the programming paradigms.	L5-6	K	Knows that programming paradigms are fundamental approaches to structuring and organizing code, defining how various components of a program interact with each other. Each paradigm represents a different way of thinking about program design, and different programming languages often support one or more paradigms. Knows to list main paradigms and their characteristics: - Imperative programming. - Functional programming. - Object-Oriented programming (OOP). - Procedural programming. - Logical programming. - Event-Driven programming. - Concurrent and Parallel programming. - Structured programming.

39. Explain the concepts for imperative procedural programming	L6	K	Knows that imperative procedural programming is a programming paradigm that focuses on specifying a sequence of steps or procedures that a program should follow to perform a task.
40. Explain the concepts for Object-Oriented Programming (OOP)	L6	K	Knows that Object-Oriented Programming (OOP) is a programming paradigm that focuses on the use of objects, classes, and methods to structure code and solve problems.
41. Explain the concepts for declarative logic	L6	K	Knows that declarative logic programming emphasizes expressing relationships and facts without specifying control flow.
42. Explain the concepts for declarative functional	L6	K	Knows that declarative functional programming centers on expressing computation as mathematical functions without relying on mutable state or side effects.
43. List the techniques for concurrency	L6	K	Knows that concurrency is the ability of a program to manage and execute multiple tasks simultaneously, improving efficiency and responsiveness. List and explain the techniques for concurrency: - multi-threading. - multi-processing.
44. Carry out concurrency	L6	S	Can identify independent tasks and determine which parts of the program can run concurrently without interfering with each other. Can use high-level concurrency abstractions that simplify managing threads and processes.
45. Testing and debugging	L6	K	Knows that testing and debugging are critical phases in software development that help ensure the quality, reliability, and correctness of a program. These processes involve identifying and fixing errors, defects, and unexpected behaviors in the code.
46. Debug an application	L6	S	Knows how to apply debugging techniques:

			- Print statements. - Interactive debugging. - Logging using libraries to record information about program execution. - Code review. - Create a test case that reproduces the problem.
47. Test an application	L7	S	Knows how to test the functionalities on an application accordingly with specification. Can perform several types of testing: - Unit testing. - Integration testing. - Functional testing. - Regression testing. - Performance testing. - Security testing.
48. Testing and debugging	L6	A	Is open to engage in testing and debugging having in mind that these processes are iterative and require attention to detail, patience, and systematic approaches.
49. Explain the differences between compiled and interpreted (scripting)	L6	K	Knows to compiled and interpreted languages are two different approaches to executing code in programming. Knows that in compiled languages, source code is transformed into machine code or an intermediate representation by a compiler before execution. Knows that in interpreted languages, the source code is executed line-by-line or statement-by-statement by an interpreter.
50. Characteristics of compiled language	L6	K	Knows that the performance of a compiled language is better because the code is translated directly into machine code, which is efficient and optimized for the target hardware, but is usually less portable as it is specific to the architecture and operating system for which it was compiled.

51. Characteristics of interpreted language	L6	S	Knows that interpreted languages generally have slower performance than compiled languages because of the real-time translation overhead, but are usually more portable as the code is not tied to a specific platform or architecture.
52. Is aware of typed of the programming language	L4	K	Knows that programming languages are statically typed or dynamically typed. Knows that in statically typed languages, the data types are known at compile-time, and type checking is performed during the compilation phase. Knows that in dynamically typed languages, data types are determined at runtime.
53. Explain the concepts for markup languages	L6	K	Knows that markup languages are a way of annotating text to define its structure, presentation, or behavior within a document. Know that markup languages are based on markup tags/elements, attributes, nesting, validation, interoperability across different platforms, devices, and software.
54. Transformation and styling of markup languages	L6	S	Knows how to apply transformation technologies to styling/presenting the structure (markup) content.

COMPETENCE AREA: 3. DIGITAL CONTENT CREATION			
COMPETENCE: 3.4 PROGRAMMING			
HIGHLY SPECIALISED			
Learning Outcome	Level	K – S - A	Example
55. Establish statements and requirements for complex problems with limited definition	L7	A	Ask critical questions related to application domain to find out problem statement or requirements that are not well-defined or clear-cut. Collaborate with domain experts.
56. Design solutions to solve complex problems with many interacting factors	L8	S	Knows how to combine a systematic approach with a strategical one, how to use domain expertise, technical skills, data analysis, modeling, collaboration, and a willingness to adapt and iterate as the understanding of the problem evolves.
57. Designs the implementation of complex computer systems	L7	S	Is able to design and integrate hardware, software, and networking components to create a functional and secure system. Is able to select the target platform, languages, or frameworks for development, manage the personal resources (time and competences), considering the ecosystem and libraries, security features, long-term support and maintenance of the language, integration with existing systems, budget and licensing.
58. Supervises the implementation of complex computer systems	L7	A	Is careful that successful applications are built using various languages and technologies. Keep an open mind and be willing to adapt as the project progresses and requirements evolve.
59. Supervise compliance with specifications	L7	A	Is careful to following of the exact specifications, to implementing the best practices on developing the code, to project structures, version control, and other concepts that help to finalize a sophisticated and maintainable

			application.
60. Implement project management methods	L8	S	Knows how to manage a software project following the planning, executing, and delivering steps, respecting team management principles, risk management and quality control.
61. Guide others in the analysis and development of applications	L7	A	Integrate your knowledge and actively contribute to professional practices, empowering your team and organization to excel in their projects. Considers continuous learning, mentorship, collaborative projects, create documentation, develop templates, develop coding standards, design reusable patterns.
62. Propose new ideas and processes to the field	L8	A	Encourage everyone to introduce innovative concepts, strategies, methods, or approaches that can bring positive changes, advancements, or efficiencies to helps organizations remain competitive and adaptable in a rapidly changing environment.

Project Coordinator:

EdiTC EDUCATION & INFORMATION TECHNOLOGY CENTRE

Partners:

DIMITRA educational organization

UNIVERSITAS GALATIENSIS

CYPRUS COMPUTER SOCIETY

UNIVERSITÀ TELEMATICA INTERNAZIONALE UNINETTUNO

MMC Mediterranean Management Centre

MTU Ollscoil Teicneolaíochta na Mumhan Munster Technological University

DSW DIGITAL SKILLS WALLET

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.