|  |
| --- |
| **FURTHER EDUCATION AND TRAINING CERTIFICATE: INFORMATION TECHNOLOGY: SYSTEMS DEVELOPMENT**  **ID 78965 LEVEL 4 – CREDITS 165** |
| **LEARNER WORKBOOK**  **SAQA: 14910**  **APPLY THE PRINCIPLES OF COMPUTER PROGRAMMING** |

**Learner Information:**

|  |  |
| --- | --- |
| **Details** | **Please Complete this Section** |
| Name & Surname: | Mila Mihlali Ngewu |
| Organisation: | Loomee Group |
| Unit/Dept: | ICT |
| Facilitator Name: | Anneline Nombeko |
| Date Started: | 18/09/2023 |
| Date of Completion: | 18/10/2023 |

**Copyright**

All rights reserved. The copyright of this document, its previous editions and any annexures thereto, is protected and expressly reserved. No part of this document may be reproduced, stored in a retrievable system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior permission.

**OVERVIEW**

|  |  |  |  |
| --- | --- | --- | --- |
| **About the Learner Workbook** | This Learner Exercise Workbook has been designed and developed to evaluate learners’ level of understanding of the  **Apply the principles of Computer Programming.** It forms part of a series of Learner Workbooks that have been developed for **FURTHER EDUCATION AND TRAINING CERTIFICATE: INFORMATION TECHNOLOGY: SYSTEMS DEVELOPMENT ID 78965 LEVEL 4 – CREDITS 165** | | |
| **Purpose** | The purpose of this Learner Exercise Workbook is to evaluate learners understanding on the specific outcomes and/or assessment criteria of the following SAQA Registered Unit Standards: | | |
| **US No** | **US Title** | **Level** | **Credits** |
| **14910** | **Apply the principles of Computer Programming** | **4** | **4** |
| **Context** | This assessment represents the Formative Assessment component of the **FURTHER EDUCATION AND TRAINING CERTIFICATE: INFORMATION TECHNOLOGY: SYSTEMS DEVELOPMENT ID 78965 LEVEL 4 – CREDITS 165** and should be completed in the classroom/training room. | | |
| **Resources** | The following are resources needed for this assessment:   1. Learner Guide; and 2. Assessment Preparation. | | |
| **Instructions to Facilitators** | Facilitators will be required to:   * Explain the completion of the workbook to each learner; and * Interview the learner on similar questions, should he/she not be able to write. | | |
| **Instructions to Learners** | Learners will be required to:   * Complete the workbook as per the instructions; * Ensure that all questions are completed; * Ensure that the completion of the workbook is their own work; * Ensure that all annexure are attached to the workbook and clearly referred to; | | |
| **Assessment Time** | Learners are required to complete this assessment within the allocated time frame of.... hours. | | |
| **Total Mark** | This formative assessment carries a total mark of **\_ points**. In order to meet the pass mark, learners are required to achieve a minimum of **80%** of the total marks. | | |
| **Equipment** | Learners are required to have the following equipment in order to complete this workbook:   * Pen and Pencil; * Ruler; and * Exam Pad – for additional paper. | | |

# **GENERAL INFORMATION**

|  |  |  |
| --- | --- | --- |
| **LEARNER DETAILS** | | |
| **Learner Full Names** | **Mila Mihlali Ngewu** | |
| **Learner ID No.:** | **9909106615084** | |
| **Organisation:** | **Loomee Group** | |
| **Unit/Dept:** | **ICT** | |
| **Contact Details:** | **Telephone /Cell Numbers:** | **Email Address:** |
| **0823655804** | **ngewumila007@gmail.com** |
| **WORKSHOP DETAILS** | | |
| **Workshop Venue:** | **NMB iHUB** | |
| **Facilitator Name:** | **Anneline Nombeko** | |
| **Date Started:** | **09/09/2023** | |
| **Date Completed:** | **10/09/2023** | |

# **ASSESSMENT PREPARATION CHECKLIST**

|  |  |  |  |
| --- | --- | --- | --- |
| **DESCRIPTION** | **YES** | **NO** | **COMMENTS/CONTINGENCY** |
| This assessment is a formative assessment and it is based on the outlined unit standard/s for the **Apply the principles of Computer Programming** module. | ✓ |  |  |
| Your assessment evidence for **Apply the principles of Computer Programming** module needs to be submitted on....... (day) of...............(month)...........(year) at the following address/place................................................................ | ✓ |  |  |
| You will be assessed based on the outlined Unit Standards. The assessment activities are linked to specific outcomes/assessment criteria of the outlined Unit Standards. | ✓ |  |  |
| To determine your competence level, the following are the methods to be used for this assessment:   1. ..................................................... 2. ..................................................... | ✓ |  |  |
| To be declared competent on **Apply the principles of Computer Programming** module (formative assessment), you should have obtained at least 80% of the total mark of this assessment. | ✓ |  |  |
| You will be provided with detailed feedback on your performance of this assessment as follows:   1. Written Feedback 2. Verbal Feedback | ✓ |  |  |
| Should you be declared “not yet competent” on this assessment, you will be entitled for re-assessment opportunity/ies. | ✓ |  |  |
| You will be required to re-submit evidence (only for areas) you were declared not yet competent. A date for re-submission will be agreed with the assessor. | ✓ |  |  |
| You will be entitled to lodge an appeal should you not be satisfied with the assessment decision of your assessment. | ✓ |  |  |
| You will be required to provide the assessor feedback on assessment procedure – this is to assist in improving the assessment practices. | ✓ |  |  |
| Your results of assessment and portfolio of evidence information will not be provided to any person without your written consent. | ✓ |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Learner’s Declaration** | | | |
| I Mila Mihlali Ngewu herewith declare that I am ready for the assessment, that we have reviewed the assessment preparation and plan, I understand the assessment process and I am happy that the assessment will be conducted in a fair manner. | | | |
| **Learner Signature:** | **Date:** | **Facilitator Signature:** | **Date:** |
|  | **09/09/2023** |  | **10/09/2023** |

Learning Unit1

**UNIT STANDARD NUMBER :** 14910

**Apply the principles of Computer Programming**

**LEVEL ON THE NQF :** 4

**CREDITS :** 4

**FIELD :** Physical, Mathematical, Computer and Life Sciences

**SUB FIELD :** Construction Information Technology and Computer Sciences

|  |  |
| --- | --- |
| **PURPOSE:** | This unit standard is intended:  to demonstrate the application of knowledge of the areas covered  for those working in, or entering the workplace in the area of systems development.  People credited with this unit standard are able to:  apply fundamental principles of problem analysis  demonstrate an understanding of different data representations used in computer programs  demonstrate an understanding of fundamental programming principles  demonstrate an understanding of high level programming language concepts  The performance of all elements is to a standard that allows for further learning in this area. |
| **LEARNING ASSUMED TO BE IN PLACE:** | |
| The credit value of this unit is based on a person having the prior knowledge and skills to:  demonstrate an understanding of fundamental mathematics and English (at least NQF level 2)  demonstrate PC competency skills (End User Computing unit standards)  describe the principles of Computer Programming. | |

|  |
| --- |
| **SESSION 1.**  **Operate computer programming development tools.** |
| **Learning Outcomes** |
| * 1. The operation demonstrates the use of the editor of the development tools to produce program source code. * 2. The operation includes the use of the syntax checker of the tools to check for syntax errors. * 3. The operation uses the tool to compile the program source code produced. |

|  |  |
| --- | --- |
| http://3.bp.blogspot.com/_0EodaYtqevU/TMun5XOj03I/AAAAAAAAAIU/lzrnWelQjgc/s1600/group-discussion.jpg | **Answer the following questions according to the instructions provided** |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **1** | **Define text editor and syntax** | **5** |

A text editor is a software application or program that is used for creating, editing, and manipulating plain text files. Text editors are commonly employed by programmers, writers, web developers, and anyone who needs to work with text-based documents.

Syntax refers to the set of rules and conventions that dictate how words and symbols are combined to form valid statements or expressions in a programming language or markup language. It defines the structure and meaning of code or text in a given context.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **2** | **Demonstrate understanding of how to operate computer programming development tools (syntax, editor or library function)** | **6** |

In this example, we'll create a Python program to calculate the factorial of a number using a code editor, Python's syntax, and the math library.

# Import the math library to use the factorial function

import math

# Input: Get the number from the user

num = int(input("Enter a number: "))

# Calculate the factorial using the math library function

factorial = math.factorial(num)

# Output: Display the result

print(f"The factorial of {num} is {factorial}")

|  |
| --- |
| **SESSION 2.**  **Demonstrate an understanding of different data representations used in computer programs.** |
| **Learning Outcomes** |
| * 1. The demonstration applies different number conversion techniques between data types (at least 2). * 2. The demonstration compares different logical data types (at least 3) in a language of choice (incl. pseudo code). * 3. The demonstration differentiates between different internal representations of data types (in ASCII). * 4. The demonstration distinguishes between different logical operators (at least 2). |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **3** | **Distinguish between different internal representations of data types (in ASCII).** | **10** |

ASCII (American Standard Code for Information Interchange) is a character encoding standard that represents text characters as numeric values. In ASCII, each character is assigned a unique numeric code between 0 and 127.

1. **Character Data Type (char):**

Representation: In most programming languages, a char data type is used to represent individual characters in ASCII.

**Usage:** char variables store a single ASCII character, and they are commonly used for processing and manipulating individual characters within strings or text.

1. **Integer Data Type (int or byte):**

Representation: In some cases, ASCII characters can be internally represented as integers (e.g., int or byte) that correspond to their ASCII values. For example, 'A' is represented as 65, 'B' as 66, and so on.

**Usage:** This representation is useful when you need to perform arithmetic operations or comparisons involving ASCII characters. For example, you can add or subtract ASCII values to manipulate characters.

1. **Binary Data Type:**

Representation: In some low-level programming languages or when working directly with binary data, ASCII characters can be represented as binary values, which are sequences of 0s and 1s. For example, 'A' is represented as 01000001 in binary.

**Usage:** This representation is useful when working with raw binary data or when encoding/decoding data for transmission or storage.

1. **String Data Type:**

Representation: In higher-level programming languages, strings are used to represent sequences of characters. Each character in the string is stored as a separate element in an array-like structure, where each element corresponds to the ASCII value of a character.

**Usage:** Strings are used to handle text and character sequences more conveniently than using individual char variables. You can manipulate and process entire words or sentences using string functions and methods.

1. **Hexadecimal Data Type:**

Representation: Hexadecimal (hex) notation represents ASCII characters as hexadecimal values. For example, 'A' is represented as 0x41 in hexadecimal.

**Usage:** Hexadecimal representation is often used when displaying ASCII values in a more human-readable form, especially in debugging or memory inspection.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **4** | **Demonstrate understanding of different logical data types (at least 3) in a language of choice (incl. pseudo code).** | **5** |

1. **Boolean Type (bool):**

Description: The bool type represents logical values, which can be either True or False. It is used for conditional statements and comparisons.

**Pseudo Code Example:**

is\_raining = True

if is\_raining:

wear\_raincoat()

else:

wear\_sunglasses()

1. **Comparison Operators:**

Description: Python provides several comparison operators that return boolean values when comparing two values or expressions. These operators include == (equal), != (not equal), < (less than), > (greater than), <= (less than or equal), and >= (greater than or equal).

**Pseudo Code Example:**

user\_age = 25

is\_adult = user\_age >= 18

1. **Logical Operators (and, or, not):**

Description: Logical operators are used to combine or modify boolean values. and returns True if both operands are True, or returns True if at least one operand is True, and not negates a boolean value.

**Pseudo Code Example:**

is\_sunny = True

is\_warm = False

should\_go\_outside = is\_sunny and not is\_warm

|  |
| --- |
| **SESSION 3.**  **Demonstrate an understanding of fundamental programming principles.** |
| **Learning Outcomes** |
| * 1. The demonstration illustrates the differences between the various algorithmic structures of programming languages, using a language of choice (incl. Pseudo code). * 2. The demonstration compares good & bad program documentation principles (at least 3), using a language of choice (incl. Pseudo code) where needed. * 3. The demonstration illustrates good programming quality assurance principles. |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **5** | **Describe various algorithmic structures of programming languages, using a language of choice (incl. Pseudo code).** | **8** |

**Sequential Structure:**

Sequential structure represents the default flow of a program where statements are executed one after another in the order they appear.

**Pseudo-Code Example:**

Set x to 5

Set y to 10

Set result to x + y

Display result

**Conditional Structure (if-else):**

Description: Conditional structures allow you to execute different blocks of code based on specified conditions.

**Pseudo-Code Example:**

Input age

If age >= 18

Display "You are an adult."

Else

Display "You are not an adult."

**Function/Procedure Structure:**

Description: Functions or procedures allow you to define reusable blocks of code that can be called with different inputs.

**Pseudo-Code Example:**

Function add\_numbers(x, y)

Set result to x + y

Return result

End Function

Set sum\_result to call add\_numbers with arguments 5 and 10

**Branching Structure (switch-case, not available in Python):**

Description: Branching structures allow you to choose from multiple execution paths based on the value of a variable or expression.

**Pseudo-Code Example:**

Input choice

Switch choice

Case 1: Display "Option 1 selected"

Case 2: Display "Option 2 selected"

Case 3: Display "Option 3 selected"

Default: Display "Invalid choice"

End Switch

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **6** | **Describe program documentation principles** | **5** |

**Clarity and Simplicity:**

Documentation should be clear, concise, and easy to understand. Use plain language and avoid unnecessary jargon or technical terms that might confuse the reader.

**Consistency:**

Maintain a consistent style and format throughout your documentation. This includes consistent naming conventions, formatting (e.g., indentation, headings, and code blocks), and terminology.

**Comments in Code:**

Include comments within the code to explain complex logic, algorithms, or any non-obvious parts of the code. Comments should clarify why certain code was written and how it works.

|  |
| --- |
| **SESSION 4.**  **Demonstrate an understanding of high level programming language concepts.** |
| **Learning Outcomes** |
| * 1. The demonstration explains what is understood by constants and variables. * 2. The demonstration illustrates the concepts of operators and expressions. * 3. The demonstration illustrates different modular programming features and variable passing. * 4. The demonstration applies different debugging techniques. |

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **7** | **Define constants and variables.** | **5** |

Constants are values or data that do not change during the execution of a program. They are fixed and immutable.

Variables:

Example:

PI = 3.14159

MAX\_VALUE = 100

Definition: Variables are symbols or identifiers used to represent data that can change or vary during the execution of a program.

Example:

age = 25

name = "John"

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **8** | **List advantages of Modular Programming:** | **5** |

**Ease of Debugging and Testing:**

Smaller modules are easier to test and debug since issues can be isolated to specific components. This simplifies the process of identifying and fixing problems.

**Collaboration and Teamwork:**

Modular programming facilitates collaboration among developers because they can work on different modules independently. This leads to better team efficiency.

**Scalability:**

As project requirements evolve or new features are added, modular code can be extended by adding or modifying modules, which simplifies the process of scaling the software.

**Improved Code Organization:**

Modules help organize code logically, making it more structured and easier to navigate. This improves code readability and maintainability.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Questions Description** | **Mark** |
| **9** | **Define Debugging.** | **4** |

Debugging is the process of identifying, isolating, and fixing errors, bugs, or defects in a software program or system to make it work as intended. It is a crucial and routine task in software development, quality assurance, and maintenance. Debugging aims to ensure that the software operates correctly, produces the expected results, and behaves as designed.