HyperionDev

Curriculum title	Python Programmer
Curriculum code	900221-000-00-00
Module code	900221-000-00-PM-05
NQF level	4
Credit(s)	40
Quality assurance functionary	QCTO - Quality Council for Trades and Occupations
Originator	MICT SETA
Qualification type	Skills Programme

Python Programmer

Learner Practical Skills Workbook

Name	
Contact Address	
Telephone (H)	
Telephone (W)	
Cellular	
Email	

Table of contents

Practical Skills Module Specifications

List of Practical Skill Module Specifications

900221-000-00-PM-01	Install computer software and hardware	L4	Cr3
900221-000-00-PM-02	Troubleshoot computer and network faults	L4	Cr8
900221-000-00-PM-03	Maintain computer and network security	L4	Cr6
900221-000-00-PM-04	Provide support to end Users	L4	Cr8
900221-000-00-PM-05	Getting Started with REST API and GUI	L4	Cr4
900221-000-00-PM-06	Use Cases with Python	L4	Cr11

You will demonstrate your competence in the knowledge topics and achievement of the assessment criteria through a process of continuous assessment (evaluation of your progress throughout the skills programme). These assessments involve interpreting evidence of your ability to perform specific tasks.

During the skills programme, you will complete various procedures and tasks that will be assessed to confirm your competence. This workbook includes formative assessment activities (which help you to assess your own learning and identify your strong and weak areas) and summative assessments (which assess your learning and competencies gained at the end of particular learning areas).

The formative assessment activities can be completed in groups, pairs, or on your own.

The **summative assessments** must always be **completed on your own**.

The activities and assessments in this workbook must be submitted to the facilitator when you have completed them. They will be added to your portfolio of evidence (PoE), which will be signed by your facilitator as evidence that you have successfully performed these tasks.

Pay close attention to your facilitator's instructions and ensure you complete the activities within the given time.

Provider Programme Accreditation Criteria

Physical Requirements:

- Valid licensed software and application, including OS
- Internet connection and hardware availability
- Examples and information specified in the scope statement and all the case studies, scenarios and access to hardware and software implied in the scope statements of the modules
- Remote learners: Provider must provide business IT simulation system (e.g. invoice processing)

Human Resource Requirements:

- Qualification of lecturer (SME):
    - NQF 5 industry recognised qualification with 1 year relevant experience
- Assessors and moderators: accredited by the MICT SETA

Legal Requirements:

- Legal (product) licences to use the software for learning and training
- OHS compliance certificate
- Ethical clearance (where necessary)

Exemptions:

- None, but the module can be achieved through RPL

Purpose of Practical Skills: Module 5

Intermediate Programming with Python, NQF 4

The focus of the learning in this Module is on providing the learner with an opportunity to utilise the REST API and GUI functions in Python programming

The learner will be required to:

- PM-05-PS01: Start using API with Python
- PM-05-PS02: Get started with GUI platform

Practical Skills Module 5: Topic 1

Topic Code	PM-05-PS01
Topic	Start using API with Python

Scope of Practical Skill

Given an applicable instruction and access to a learning platform, the learner must be able to:

- PA0101 Get an API key
- PA0102 Test API endpoints
- PA0103 Retrieve information – GET
- PA0104 Add new data – POST
- PA0105 Change existing information – PUT
- PA0106 Delete existing information – DELETE
- PA0107 View the status codes of the response
- PA0108 Generate a Python snippet

Applied Knowledge

- AK0101 Concept, definition and functions of REST API

Internal Assessment Criteria

- IAC0101 Expected results with Python REST API are achieved

Resources:

Knowledge	**Information from Python Programmer Curriculum**
National Curriculum Framework	900221-000-00-PM-05

Practical task

Follow the facilitator's instructions to complete the following activities:

Scenario:
A **weather monitoring application** interacts with a weather API to fetch real-time weather data. In a real-world application, user preferences (such as preferred city, temperature units, contact method etc.) would typically be stored and managed on your own backend server, not through the weather API. Since we're not building a backend in this task, you will simulate user preference actions using a **mock API** such as JSONPlaceholder, which mimics real API behavior.

In this task, you will:
- Use the weather API (e.g., OpenWeatherMap) for retrieving weather data.
- Simulate adding, updating, and deleting user preferences using a mock API.
- Handle API response status codes appropriately in your application.
- Generate Python code snippets to automate these tasks.

Create Python file called **weather_api.py** and complete the following:

Step 1: Get an API key

- Sign up for a weather API service (e.g., **OpenWeatherMap**) and generate an API key to authenticate requests.

Step 2: Test API endpoints

- Verify that the weather API is accessible and your API key is working by making a simple GET request with a test city (for example: London).
- You can use parameters like **"q"** (representing the city name) and **"appid**" (representing your API key) to retrieve location information from the API.
- Explore other parameters you can use to access more information from this **resource**.

Step 3: Retrieve information – GET

- Fetch current weather information for any city (for example: New York).

Step 4: Add new data – POST

- Simulate saving a new user's weather notification preferences using the mock API.

Step 5: Change existing information – PUT

- Simulate updating the notification preferences for a user using the mock API.

Step 6: Delete existing information – DELETE

- Simulate removing a user's notification preferences using the mock API.

Step 7: View the status codes of the response

- Check and handle response status codes after each API interaction to confirm success or manage errors.

Step 8: Generate a Python snippet

- Generate reusable Python code to automate interactions with the weather API.

Your facilitator will complete the following evaluation checklist:

Check that the following is accomplished:

Item	**Checked (Yes=5 No=0)**	**Comment: where did you find the evidence?**
PA0101 Get an API key		
PA0102 Test API endpoints		
PA0103 Retrieve information – GET		
PA0104 Add new data – POST		
PA0105 Change existing information – PUT		
PA0106 Delete existing information – DELETE		

PA0107 View the status codes of the response		
PA0108 Generate a Python snippet		
Name of member		
Signature		
Date		
	Total	**/40**

Take note

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Practical Skills Module 5: Topic 2

Topic Code	PM-05-PS02
Topic	Get started with GUI platform

Scope of Practical Skill

Given a range the learner must be able to:

- PA0201 Import selected module
- PA0202 Create the GUI application main window
- PA0203 Add widgets to the GUI application
- PA0204 Handle user interactions with the event loop
- PA0205 Structure the GUI layout

Applied Knowledge

- AK0201 Concept, definition and functions of GUI

Internal Assessment Criteria

- IAC0201 Expected results with Python GUI are achieved

Resources:

Knowledge	**Information from Python Programmer Curriculum**
National Curriculum Framework	900221-000-00-PM-05

Practical task

Follow the facilitator's instructions to complete the following activities:

A **student management application** allows users to input and display student data (like name and age) through a graphical user interface (GUI). The application uses Python's PyQt library to create the GUI, manage user input, and respond to events like button clicks.

Create a Python file called **student_management.py** and complete the following:

Step 1: Import selected module

- Import the **PyQt5.QtWidgets** module to access the necessary classes for building the GUI.

Step 2: Create the GUI application main window

- Create the main window for the student management application.
- **QApplication** can be used to initialize the app and **QWidget** can be used to create the main window.

Step 3: Add widgets to the GUI application

- Add input fields, labels, and buttons to collect and display student information.
- You can use widgets such as **QLabel** (for labels), **QLineEdit** (for text input), and **QPushButton** (for user actions).

Step 4: Handle user interactions with the event loop

- Respond to user actions like button clicks to save and display student data.
- You can set up this loop using **app.exec_()** to make the app interactive.

Step 5: Structure the GUI layout

- Combine all widgets to create a complete GUI for managing student data.

- You can use layout managers like **QVBoxLayout** or **QGridLayout** to arrange your widgets neatly on the main window.

Your facilitator will complete the following evaluation checklist:

Check that the following is accomplished:

Item	**Checked (Yes=5 No=0)**	**Comment: where did you find the evidence?**
PA0201 Import selected module		
PA0202 Create the GUI application main window		
PA0203 Add widgets to the GUI application		
PA0204 Handle user interactions with the event loop		
PA0205 Structure the GUI layout		
Name of member		
Signature		
Date		
Total		**/25**

Take note

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Requirements met **49+/65**		**Total**	**/65**
Requirements not met **Under 49/65**			
Facilitator signature		**Date**	

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.