MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	Python Programming			**Module Delivery**	
Module Type	**Core**			☒**Theory** ☒**Lecture** ☒**Lab** ☐**Tutorial** ☐**Practical** ☐**Seminar**	
Module Code	**MU010602204**				
ECTS Credits	**5**				
SWL (hr/sem)	**125**				
Module Level	UGII		**Semester of Delivery**	4	
Administering Department	Computer Science		**College**	College of Science	
Module Leader	Dr. Muhanad Tahrir Younis		**e-mail**	mty@uomustansiriyah.edu.iq	
Module Leader's Acad. Title	Asst. Prof.		**Module Leader's Qualification**		
Module Tutor	DR. Ayman Majed		**e-mail**	aymanmajid@uomustansiriyah.edu.iq	
Peer Reviewer Name	Lecturer Firas Ali Hashim		**e-mail**	firas70@uomustansiriyah.edu.iq	
Scientific Committee Approval Date	22 / 4 /2024		**Version Number**	1.0	

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	Programming Fundamentals I (1) Programming Fundamentals II (1)	**Semester**	(1) (2)
Co-requisites module	None	**Semester**	

Module Aims, Learning Outcomes and Indicative Contents

أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives أهداف المادة الدراسية	The **Python Programming** module is a comprehensive introduction to Python, one of the most popular and versatile programming languages in the world today. Throughout the course, students will be guided through the fundamentals of Python, starting from basic syntax and data types, progressing through more complex topics such as object-oriented programming (OOP), and culminating in advanced concepts like file handling, error handling, and working with libraries and modules. Each topic is introduced with clear explanations, practical examples, and hands-on coding exercises to reinforce understanding. Python's simplicity and readability make it an ideal first language for aspiring programmers, while its powerful capabilities have established it as a go-to language for professionals in various fields, including data science, artificial intelligence, and more. This course emphasizes practical problem-solving and real-world application, ensuring that students not only learn the syntax and structure of Python but also how to think computationally and develop efficient, maintainable code. The course also includes a series of projects that allow students to apply what they have learned to more substantial programming tasks. These projects range from simple command-line tools to more complex applications that involve data manipulation and interaction with external libraries. By the end of the course, students will have developed a solid foundation in Python programming and will be able to tackle intermediate and advanced Python tasks with confidence.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	By the end of the course, students will be able to: 1. **Master Python Fundamentals:** Understand the core syntax, data types, and control structures of Python. 2. **Build Problem-Solving Skills:** Develop the ability to break down problems and implement solutions using Python's functions, loops, and conditions. 3. **Apply Object-Oriented Programming:** Understand and apply object-oriented programming principles to create reusable and modular code. 4. **Work with Data and Files:** Efficiently handle data and file operations in Python. 5. **Utilize Python Libraries:** Leverage Python's standard libraries and third-party modules to extend functionality and perform data analysis using Python libraries such as Pandas and NumPy. 6. **Develop Real-World Applications:** Design and implement Python applications with graphical user interfaces. 7. **Prepare for Advanced Learning:** Develop a foundation for advanced topics in programming and software development.
Indicative Contents المحتويات الإرشادية	Indicative content includes the following. **UNIT I: [**5 hrs.] **Introduction to Python**: This unit introduces students to the fundamentals of programming and Python. It covers the basics of setting up the programming environment, writing simple Python programs, and understanding the Python

	interpreter and syntax. **UNIT-II**: [10 hrs.] **Python Basics: Variables, Data Types, and Operators:** This unit covers Python's basic building blocks, including variables, data types, and operators. Students will learn how to handle different types of data and perform basic operations. **UNIT-III**: [10 hrs.] **Control Structures: Conditional Statements and Loops:** This unit introduces control flow in Python, including conditional statements and looping structures. Students will learn to manage the flow of their programs using if, else, while, and for statements. **UNIT-IV:** [10 hrs.] **Functions and Modular Programming:** This unit focuses on defining and using functions in Python. It covers function arguments, return values, and the concept of modular programming by creating and importing modules. **UNIT-V** [10 hrs.] **Data Structures: Lists, Tuples, Dictionaries, and Sets:** This unit explores Python's built-in data structures, including lists, tuples, dictionaries, and sets. Students will learn how to store, access, and manipulate data using these structures. **UNIT-VI** [5 hrs.] **Working with Strings:** This unit covers string manipulation and formatting in Python. Students will learn how to handle and process strings, including using regular expressions for pattern matching. **UNIT-VII** [5 hrs.] **File Handling:** This unit covers string manipulation and formatting in Python. Students will learn how to handle and process strings, including using regular expressions for pattern matching. **UNIT-VIII** [10 hrs.] **Object-Oriented Programming (OOP):** This unit introduces the concepts of object-oriented programming, including classes, objects, inheritance, polymorphism, and encapsulation. Students will learn to define classes, create objects, and use class attributes and methods. **UNIT-IX** [5 hrs.] **Working with Libraries and Modules:** This unit covers the use of Python libraries and modules, such as Numpy, Pandas, Matplotlib and Tkinter. Students will learn to use standard libraries, third-party libraries, and create custom modules for extended functionality.

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	

Strategies	The following Learning and Teaching strategies are designed to create an engaging, interactive, and supportive learning environment that caters to different learning styles while ensuring that students gain a deep and practical understanding of Python programming.
	
	Learning and Teaching Strategies
	
	1. **Lectures and Demonstrations**
	   - **Description:** Each session will begin with a lecture introducing the key concepts and theoretical aspects of the topic at hand. Lectures will be supported by live coding demonstrations, where the instructor will walk through examples to illustrate how the concepts are applied in practice.
	   - **Objective:** To provide a clear and structured understanding of Python programming concepts and their real-world applications.
	2. **Hands-On Coding Exercises**
	   - **Description:** After each lecture, students will engage in hands-on coding exercises designed to reinforce the material covered. These exercises will range from basic syntax practice to more complex problem-solving tasks.
	   - **Objective:** To ensure students apply what they have learned immediately, reinforcing their understanding and building confidence in their programming skills.
	3. **Collaborative Learning and Group Projects**
	   - **Description:** Students will work in small groups on select assignments and projects, encouraging collaboration and peer learning. Group projects will simulate real-world scenarios where teamwork and collaboration are essential.
	   - **Objective:** To develop teamwork and communication skills while deepening understanding through peer discussion and collaboration.
	4. **Interactive Labs**
	   - **Description:** Dedicated lab sessions will be provided where students can experiment with Python programming in a guided environment. The labs will focus on specific tasks, such as debugging, file handling, or working with libraries.
	   - **Objective:** To offer students a space for experimentation and deeper exploration of complex topics, with instructor support available for troubleshooting and guidance.
	5. **Regular Quizzes and Assessments**
	   - **Description:** Quizzes will be administered regularly to assess students' understanding of the material. These assessments will include both theoretical questions and practical coding tasks.
	   - **Objective:** To track progress and identify areas where students may need additional support, ensuring continuous learning and improvement.
	6. **Final Project**
	   - **Description:** The course will culminate in a final project where

	students will develop a complete Python application. This project will require the integration of all the concepts learned throughout the course. ○ **Objective:** To provide a capstone experience where students demonstrate their ability to apply Python programming skills to a real-world problem, from planning and development to testing and presentation. 7. **Flipped Classroom Approach** ○ **Description:** For certain topics, a flipped classroom approach will be used, where students are provided with reading materials, videos, or tutorials to review before class. Classroom time will then be used for discussion, problem-solving, and hands-on activities related to the pre-class materials. ○ **Objective:** To maximize classroom interaction and hands-on learning, while encouraging students to engage with the material at their own pace before applying it in class. 8. **Peer Review and Code Review Sessions** ○ **Description:** Students will participate in peer review sessions where they will critique and provide feedback on each other's code. These sessions will also include guided code reviews by the instructor to highlight best practices and common pitfalls. ○ **Objective:** To develop critical thinking and attention to detail, as well as to promote best practices in coding, such as readability, efficiency, and maintainability. 9. **Use of Online Resources and Tools** ○ **Description:** Students will be encouraged to use online platforms like GitHub for version control, Stack Overflow for problem-solving, and Python documentation for reference. Additionally, interactive coding platforms like Jupyter Notebooks may be used for in-class activities. ○ **Objective:** To familiarize students with the tools and resources used by professional Python developers, fostering independent learning and resourcefulness. 10. **Regular Feedback and One-on-One Sessions** ○ **Description:** The instructor will provide regular feedback on assignments and projects, and one-on-one sessions will be available for students who need extra help or have specific questions. ○ **Objective:** To ensure personalized support for each student, addressing individual challenges and helping them achieve their learning goals.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ أسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	79	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	5
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	46	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	3

Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	**125**

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	**Quizzes**	2	10% (10)	5 and 11	LO #1, #2
	Assignments	2	8% (8)	6 and 13	LO #1, #2, #3, #4
	Projects / Lab.	1	14% (14)	Continuous	All
	Report	1	8% (8)		All
Summative assessment	**Midterm Exam**	2hr	10% (10)	8	LO #1, #2, #3
	Final Exam	4hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	**Introduction to Python**: Overview of Python: history, features, and applications, Installing Python and setting up the development environment, Writing and executing your first Python program, Understanding the Python interpreter and syntax basics
Week 2	**Python Basics: Variables, Data Types, and Operators:** Variables and assignment in Python Data types: integers, floats, strings, and Booleans, Basic operators: arithmetic.
Week 3	**Python Basics: Variables, Data Types, and Operators:** Basic operators: comparison, logical, and assignment, Type conversion and type checking, Input and output functions.
Week 4	**Control Structures-Conditional Statements**: if, if else, Nested if..else, if-elif-else, and Ternary Expression
Week 5	**Control Structures- Loops:** Looping structures: while and for loops, Nested Loop, Loop control statements: break, continue, and pass
Week 6	**Functions and Modular Programming:** Defining and calling functions, Function arguments: positional, keyword, and default values, Returning values from functions, Scope and lifetime of variables.
Week 7	**Functions and Modular Programming:** Modular programming: creating and importing modules, The concept of recursion.
Week 8	**Midterm Exam**
Week 9	**Data Structures-Lists and Tuples:** Lists: creation, indexing, slicing, and methods, Tuples:

	immutability and use cases.
Week 10	**Data Structures- Dictionaries and Set**s: Dictionaries: key-value pairs, methods, and practical applications, Sets: uniqueness, set operations, and methods, Iterating over data structures.
Week 11	**Working with Strings**: String creation and manipulation, String methods and operations, String formatting techniques, Handling special characters and escape sequences, Regular expressions for pattern matching
Week 12	**File Handling:** File operations: opening, reading, writing, and closing files, Working with text and binary files, File modes: read, write, and append, Handling file exceptions, Reading and writing CSV files**.**
Week 13	**Object-Oriented Programming (OOP):** Introduction to OOP concepts: classes and objects, Defining classes and creating objects, Class attributes and methods, The self keyword and object state**.**
Week 14	**Working with Libraries and Modules:** Introduction to Python standard libraries, Importing modules and using built-in functions, Working with third-party libraries (e.g., NumPy, Pandas, Matplotlib and Tkinter), Creating basic windows and dialogs, Adding widgets: buttons, labels, text boxes, etc.
Week 15	**Project Discussion**
Week 16	**Final exam**

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	**Lab 1:** Setting up Python on different operating systems (Windows, macOS, Linux) and writing a simple "Hello, World!" program. Exploring the Python interpreter through basic commands.
Week 2	**Lab 2:** Writing programs to explore variables, data types, and operators.
Week 3	**Lab 3:** Creating small programs to perform calculations and handle user input/output.
Week 4	**Lab 4:** Developing programs to implement conditional statements, such as calculating max number, checking odd and even numbers, and creating a basic calculator.
Week 5	**Lab 5:** Developing programs to implement loops, such as calculating factorials, checking prime numbers, and creating patterns.
Week 6	**Lab 6:** Writing functions to solve specific tasks, such as max number and sum.
Week 7	**Lab 7:** Writing functions to solve recursion tasks, and breaking down larger programs into smaller modules. Example: Implementing a recursive function to calculate sum, factorial, power, and Fibonacci series.
Week 8	**Midterm Exam**
Week 9	**Lab 8:** Implementing programs that manipulate Lists and Tuples data structures. Examples include sorting a list, checking whether an element exists within a tuple, and removing an item from a list and a tuple.
Week 10	**Lab 9:** Implementing programs that manipulate Dictionaries and Sets data structures.

	Examples include finding the intersection of sets, and using dictionaries to count word frequencies in a text.
Week 11	**Lab 10**: Building programs that perform string manipulation and pattern matching using regular expressions. Example: Validating email addresses or parsing text files for specific patterns.
Week 12	**Lab 11:** Creating programs to read from and write to files. Handling exceptions related to file operations. Example: Writing a program to read data from a CSV file, process it, and write the output to another file.
Week 13	**Lab 12:** Creating basic classes and objects. Implementing simple applications using OOP concepts such as a class for a BankAccount with methods to deposit and withdraw money.
Week 14	**Lab 13:** Developing programs that utilize various libraries and modules. Example: Using the Pandas library to manipulate and analyze datasets, or using Matplotlib for basic plotting. Building a basic GUI application using Tkinter. Example: Developing a simple calculator or a to-do list application with a graphical user interface.
Week 15	**Project Discussion**
Week 16	**Final exam**

Learning and Teaching Resources
مصادر التعلم والتدريس

	Text	Available in the Library?
Required Texts	Matthes, Eric. ***Python crash course: A hands-on, project-based introduction to programming.*** No starch press, 2023.	
Recommended Texts	1- Liang, Y. Daniel. ***Introduction to programming using Python***. Pearson, 2013. 2- McKinney, Wes. ***Python for data analysis***. O'Reilly Media, Inc., 2022.	
Websites		

Grading Scheme
مخطط الدرجات

Group	Grade	التقدير	Marks %	Definition
Success Group (50 - 100)	**A -** Excellent	امتياز	90 – 100	Outstanding Performance
	B - Very Good	جيد جدا	80 – 89	Above average with some errors
	C - Good	جيد	70 – 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 – 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 – 59	Work meets minimum criteria

Fail Group (0 – 49)	**FX –** Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.