MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Introduction to Python مقدمة في برمجةProgramming / البايثون**			**Module Delivery**	
Module Type	**Core**			☒ **Theory** ☒ **Lecture** ☒ **Lab** ☐ **Tutorial** ☐ **Practical** ☐ **Seminar**	
Module Code	**CYBT105**				
ECTS Credits	**8**				
SWL (hr/sem)	**200**				
Module Level			**Semester of Delivery**		2
Administering Department		Cybersecurity Technical Engineering / قسم الأمن تقنيات هندسة السيبراني	**College**	Technical Engineering College of Computer and AI - Kirkuk / التقنية الكلية كركوك - الاصطناعي والذكاء للحاسوب الهندسية	
Module Leader	Dr. Academic Leader		**e-mail**	leader@ntu.edu.iq	
Module Leader's Acad. Title		Assistant Professor	**Module Leader's Qualification**		Ph.D. in Computer Science / Software Engineering
Module Tutor	Tutor Name		**e-mail**	tutor@ntu.edu.iq	
Peer Reviewer Name		Reviewer Name	**e-mail**	reviewer@ntu.edu.iq	
Scientific Committee Approval Date		01/09/2024	**Version Number**		1.0

Relation with other Modules العلاقة مع المواد الدراسية الأخرى

Prerequisite module	Programming Fundamentals (CYBT103)	**Semester**	1
Co-requisites module	Python in Cybersecurity / Data Structures & Algorithms	**Semester**	3 & 4

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Aims أهداف المادة الدراسية	This module aims to provide first-year cybersecurity engineering students with foundational programming knowledge and hands-on skills in Python. It covers fundamental programming concepts including syntax, data types, control flow structures, functions, modular design, object-oriented programming (OOP) basics, file handling, exception management, standard libraries, and introductory scripting applications essential for software development and cybersecurity.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	Comprehend Python syntax, execution environment, variables, primitive data types, and operators. Implement control structures (conditional statements and iterative loops) to solve algorithmic problems. Utilize built-in data structures (lists, tuples, dictionaries, sets) for efficient data manipulation. Design modular Python programs using user-defined functions, arguments, return values, and scope rules. Apply basic Object-Oriented Programming (OOP) concepts including classes, objects, encapsulation, and inheritance. Perform file input/output operations and implement robust exception handling mechanisms. Develop Python scripts utilizing standard libraries (math, os, sys, re, random) for task automation and problem solving.
Indicative Contents المحتويات الإرشادية	The module will cover the following key areas: Introduction to Python: Environment Setup (IDLE, VS Code, PyCharm), Execution Models, Syntax, and Comments Variables, Data Types & Operators: Integers, Floats, Strings, Booleans, Type Casting, Arithmetic, Logical, and Bitwise Operators Control Flow I: Conditional Statements (if, elif, else), Nested Conditions, and Logical Expressions Control Flow II: Loop Structures (for, while), Iteration Tools, Range Function, break, continue, and pass Statements Python Data Structures I: Lists (Indexing, Slicing, Methods, List Comprehensions) and Tuples (Immutability, Operations) Python Data Structures II: Dictionaries (Key-Value Pairs, Methods) and Sets (Uniqueness, Set Operations) Functions & Modular Programming: Function Definition, Parameters, Default Values, *args, **kwargs, Scope, and Lambda Functions Python Modules & Packages: Importing Built-in Modules (math, random, datetime, os, sys) and Creating Custom Modules

	File Handling & I/O: Reading/Writing Text and CSV Files, File Modes, Context Managers (with Statement) Exception Handling: Try-Except Blocks, Raising Exceptions, Custom Exceptions, and Debugging Techniques Object-Oriented Programming (OOP) Basics: Classes, Objects, Attributes, Methods, __init__ Constructor, and Basic Inheritance Introduction to Regular Expressions & Scripting: String Manipulation, Regex Basics (re module), and Automation Utilities

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	The module will employ a combination of strategies including: Lectures: To introduce theoretical concepts of programming, syntax, control structures, and OOP paradigms. Interactive Live Coding Demonstrations: Real-time code construction and debugging during lecture sessions. Practical Laboratory Sessions: Hands-on coding exercises, algorithm implementation, and script building in Python IDEs. Applied Coding Assignments: Homework problems focusing on mathematical algorithms, string parsing, and file utilities. Tutorials: Code review sessions, logic refinement, and step-by-step problem-solving assistance. Mid-term and final exams: To evaluate theoretical syntax understanding, logical problem solving, and practical coding proficiency. Preparatory Week: To allow students time to revise coding concepts and debug projects prior to the final examination.

Student Workload (SWL) الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا			
Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	63	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	4
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	87	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	6
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	150		

Module Evaluation
تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	5% (5)	5, 10	LO #1, 3 and 5
	Assignments	2	5% (5)	2, 12	LO #2, 4 and 6
	Projects / Lab.	1	10% (10)	Continuous	LO #7
	Report	1	10% (10)	13	LO #3, 5 and 7
Summative assessment	Midterm Exam	2 hr	20% (20)	8	LO #1 - 4
	Final Exam	2hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)
المنهاج الاسبوعي النظري

	Material Covered
Week 1	Introduction to Python Programming: History, Characteristics, Interpreted vs Compiled, IDE Installation & Workflow.
Week 2	Python Basics: Variables, Dynamic Typing, Naming Conventions, Input/Output Functions (input, print, format).
Week 3	Data Types & Operators: Numbers, Booleans, Strings, Arithmetic, Comparison, Logical, Assignment, and Bitwise Operators.
Week 4	Control Flow - Decision Making: If Statements, If-Else, If-Elif-Else Ladders, Nested Conditionals, and Ternary Operators.
Week 5	Control Flow - Repetition: While Loops, For Loops, Iterating Sequences, Range Function, Break, Continue, Pass.
Week 6	Sequence Data Types - Lists: List Creation, Indexing, Slicing, List Methods (append, extend, pop, sort), List Comprehensions.
Week 7	Sequence & Mapping Types - Tuples & Dictionaries: Tuple Operations, Dictionary Key-Value Mapping, Methods, and Iteration.
Week 8	Mid-term Exam (الفصل منتصف اختبار) & Set Data Structures: Set Operations (Union, Intersection, Difference), Frozen Sets.
Week 9	Functions I: Defining Functions, Positional & Keyword Arguments, Default Parameters, Return Values, Variable Scope.

Week 10	Functions II: Advanced Arguments (*args, **kwargs), Recursion, Anonymous Functions (Lambda), Map, Filter, Reduce.
Week 11	Modules & Packages: Module Concept, Importing Built-in Modules (math, random, datetime), Creating Custom Modules.
Week 12	File Handling I: File Modes (r, w, a), Reading and Writing Text Files, Context Managers ('with' Statement).
Week 13	File Handling II & Serialization: Parsing CSV Files, Working with JSON Data, Directory Manipulation (os, sys modules).
Week 14	Exception Handling & Debugging: Syntax Errors vs Exceptions, Try-Except-Finally Blocks, Raising Exceptions, Custom Errors.
Week 15	Introduction to Object-Oriented Programming: Classes, Objects, Attributes, Methods, __init__ Constructor, Encapsulation.
Week 16	OOP Inheritance & Scripting Application: Basic Inheritance, Method Overriding, String Parsing (re Module), Course Review & Final Exam.

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	Lab 1: Python Development Environment Setup: Installing Python, VS Code / PyCharm, Running First Scripts, I/O Drills.
Week 2	Lab 2: Coding Practice on Operators & Data Types: Arithmetic Calculators, String Manipulations, and Formatting Drills.
Week 3	Lab 3: Control Flow Workshop: Writing Algorithms for Conditional Checks, Grading Systems, and Loop Iterations.
Week 4	Lab 4: Data Structures Workshop: Hands-on Exercises on Lists, Tuples, Dictionaries, Slicing, and Sorting Algorithms.
Week 5	Lab 5: Modular Coding Lab: Writing Reusable User-Defined Functions, Math Utilities, and Custom Module Imports.
Week 6	Lab 6: File I/O & Exception Lab: Building a Text/CSV File Analyzer with Robust Try-Except Error Handling.
Week 7	Lab 7: Practical Coding Assessment: Mini-Project Implementation (Class Design & Scripting), Practical Exam, & Course Review.

Learning and Teaching Resources مصادر التعلم والتدريس		
	Text	**Available in the Library?**
Required Texts	Matthes, E. (2019). "Python Crash Course: A Hands-On, Project-Based Introduction to Programming" (2nd ed.). No Starch Press.	Yes
Recommended Texts	Lutz, M. (2013). "Learning Python" (5th ed.). O'Reilly Media.	Yes
Websites	Yes	

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.