JOHN ABBOTT CEGEP/COLLEGE COURSE OUTLINE

A. GENERAL INFORMATION:

1. Program name(s) (if applicable):	Science (200.B1)T
2. Course title:	**Programming in Science**
3. Course and section number(s):	420-SN1-RE
4. Ponderation (weekly class – lab/fieldwork – homework hours):	1-2-3
5. Credits:	2
6. Competency statement and code:	0F01: Develop software programs to automate problem-solving in a scientific context. Covered: Completely
7. Prerequisite (if any):	None
8. Semester:	Fall 2024
9. Teacher name(s), (pronouns if desired):	TBD
10. Office number, phone extension (email address optional):	TBD
11. Teacher's availability:	TBD

B. INTRODUCTION (including any program-related information):

Course summary:

This course introduces structured and disciplined approaches to computer programming and problem solving in a scientific context.

Students will gain the ability to analyze a word problem and design an algorithm to solve it. This includes implementation of the algorithm in `Python`, verification of correctness and accuracy of the algorithm, and displaying results either textually or graphically.

Role and place of the course:
Programming is a required skill for almost all scientific endeavours, whether to model data or analyze data or even predict outcomes using modern AI techniques. As a result, this course is offered in the first year of the science program, to ensure students can leverage their new skills in their future studies.

Other information (links to Integrative Activity, Comprehensive Assessment, Exit Profile, certificates…):

C. COURSE OBJECTIVES:

Upon successful completion of this course, students will be able to use the `Python` programming language to:

- take a word problem and design an algorithm to solve it
    - evaluate the data that is known
    - determine what data is unknown
    - determine the process required to calculate the data
- analyze an algorithm and its associated code to evaluate correctness, and fix any errors
- use appropriate libraries or IDEs (Integrated Development Environment) to test the correctness of the algorithm/code
- obtain data from users and from structured documents
- display data using relevant plotting techniques
    - use effective sampling techniques and appropriate storage of data
    - apply filters to data to improve readability of the graphed data
- use standard science libraries such as `numpy`, `plotly`, `pandas` and others
- simulate physical models and statistical models
- implement standard algorithms in `Python`

Government Competency:
0F01 - Develop software programs to automate problem-solving in a scientific context.

Performance Criteria for the Competency as a Whole
- Appropriate use of terminology
- Correct use of required software development tools
- Observance of program development steps
- Respect for digital ethics regarding intellectual property
- Demonstration of autonomy, rigour and perseverance

	Elements of the Competency	**Performance Criteria**
0F01.1	Plan the automation for solving of a problem.	• Appropriate recognition of the concepts involved in a problem • Accurate determination of inputs and outputs • Accurate determination of the necessary processing • Coherent decomposition of the algorithm • Preparation of appropriate test cases to validate the functioning of the program
0F01.2	Code the algorithm in the programming language.	• Logical organization of instructions • Appropriate use of basic data types and arrays • Accurate use of arithmetic, relational and logical expressions • Adherence to programming language syntax and conventions • Appropriate use of libraries
0F01.3	Verify the program's operation.	• Correct tracing of program execution • Detection of operational errors • Relevance of the corrections made

D. EVALUATION PLAN

Evaluation type:	%	Tentative date:	Link to competencies/objectives/ competency elements:	√ if part of final evaluation
Test	20	Week 7-8	0F01.1, 0F01.2	
Final Exam	40	Final Exam Period	0F01.1, 0F01.2	√
Assignments (4-5)	30	Distributed over the 15-week semester	0F01.1, 0F01.2	√
Minor Assessments	10	Distributed over the 15-week semester	0F01.1, 0F01.2, 0F01.3	√
Total value:	*100%*	TBA	0F01.1, 0F01.2, 0F01.3	
Value of final eval (min 40%):	*80*			

E. COURSE CONTENT:

Analyzing and Interpreting Problems

- From a problem statement, determine the Input/Output components (What is given/What is wanted)
- Design algorithms
- Write the pseudocode for an algorithm

Tools

- PyCharm IDE
- Trace and Debug
- Online resources/documentation

Concepts required to implement solutions

- Basic Input/Output
- Data Types:
    - Numeric
    - String/Character
    - Boolean
    - Objects and calling methods
- Collections
    - Ranges, tuples, lists, sets and dictionaries
- Control structure:
    - Conditionals: if
    - Looping: for and while
- Functions

    - Creating and using functions
    - Mutable and immutable arguments.
- Libraries
    - Importing and using external libraries.
- Graphing
    - Plot data

Quality of solutions

- Type hinting
- Commenting
- Documentation
- Naming conventions
- Testing
- Edge cases

Input / Output

- Reading and writing to console and files.
- Reading and writing basic data types.
- Parsing structured data: delimited data.

Tentative schedule

Topics:	Week/Dates	Additional Info:
Intro to the IDE and other required tools	1	
Best Practices	2	
Basic Data types, Reading/Writing to Console	3	
Conditionals (& debugging)	4	
Loops & Lists (& debugging)	5	
Functions (& debugging)	6	
Functions (& debugging)	7	
Review	8	Test
Algorithm Design and Implementation	9	
Algorithm Design and Implementation	10	
Intro to numpy library	11	
More numpy	12	
Reading from a file	13	
Graphing data	14	
Review	15	

F. REQUIRED TEXTBOOKS/MATERIALS, COURSE COSTS IN ADDITION TO TEXTS

On-line resources, to be given as required.

G. INSTRUCTIONAL METHODS:

Classes will be given in a computer science lab. The class time will be split between lecture and lab-time. Overall, the proportion of lecture to lab time will be one hour lecture for every two hours lab, but this split will differ daily depending on the current material being covered.

In class assessments can be done in a group or individually.

Assignments must be individual work.

H. PROGRAM, DEPARTMENTAL/DISIPLINE, AND COURSE/SECTION POLICIES:

Policy:	Description:
Approved department attendance policy	N/A
Policy to ensure that issues relating to late submission, or resubmission, of work to be dealt with in an equitable manner	10% per day late.
Policy dealing with the expectations of classroom behaviour, including use of cell phones, laptops and other technology	Refer to college policies regarding behaviour in the classroom.
Other expectations (if applicable)	

I. COLLEGE POLICIES:

Topic:	Resource:
Student rights and responsibilities (articles 3.2 and 3.3)	Policy no. 7: IPESA- Institutional Policy on the Evaluation of Student Achievement (Version: June 12, 2019)
Changes to evaluation plan in the course outline (article 5.3)	
Religious holidays (articles 3.2.13 and 4.1)	
Cheating and plagiarism (articles 9.1 and 9.2)	
Cheating and plagiarism academic procedure and other resources	Academic Integrity: Cheating and Plagiarism Procedure (version: October 22, 2021) • **You need to log into Omnivox to access the above document** • **For PowerPoint on cheating and plagiarism** refer to the JAC Portal: My JAC Communities / Academic Council / Curriculum Validation Committee (CVC) / Course Outlines – Reference Documents / Academic Integrity

	• **For link to interactive tutorial on how to cite sources correctly**: http://citeit.ccdmd.qc.ca
Code of conduct	Policy 13: Policy on Student Conduct and Discipline Procedures (version: September 21, 2021)