# LIFE733: Python Programming – Weeks 1–5 Teaching and Assessment Plan

# Overview

This plan outlines the weekly structure and assessment strategy for the first 5 weeks of LIFE733: Introduction to Python Programming. The course is designed to introduce fundamental programming concepts to students in the biosciences through realistic and increasingly complex coding problems. The content scaffolds toward two summative assessments that are auto-markable and resilient to misuse of generative AI.

# Week 1 – Getting Started with Python: Variables, Input/Output, and Strings

Learning Objectives:  
- Write and run Python scripts in VS Code  
- Use `input()` and `print()` for user interaction  
- Store and manipulate string data  
- Apply string methods and slicing  
- Understand basic types and conversions

Formative Tasks:  
- Print user input with formatting  
- Calculate DNA string length and case transformations  
- Concatenate and reverse DNA sequences  
- Convert DNA to RNA and add a polyA tail

Setup Verification Task:  
- Create a script that prints a confirmation message and Python version  
- Used to ensure VS Code is correctly installed on MWS and personal devices

YouTube Video:  
- “Working with Strings and Input in Python”

# Week 2 – Conditionals and Mathematical Operations

Learning Objectives:  
- Use `if`, `elif`, and `else` statements  
- Compare and validate numerical inputs  
- Implement custom formulae for mean, variance, and standard deviation

Formative Tasks:  
- Determine larger of two values  
- Identify triangle types from side lengths  
- Calculate summary statistics from 5 inputs

YouTube Video:  
- “Conditionals and Calculating Statistics in Python”

# Week 3 – Lists, Loops, and Classification

Learning Objectives:  
- Use lists to store and manipulate data  
- Loop over list values with `for` loops  
- Classify and sort data based on numeric rules  
- Handle invalid or outlier values

Formative Tasks:  
- Sort exam scores into grade categories  
- Compute category means  
- Identify and report errors by index

YouTube Video:  
- “Classifying Data with Lists and Loops”

# Assessment 1 – Applied Programming Skills

Submission: 4 separate `.py` files  
Auto-Marked: Yes (print output only)

Part 1A – mRNA String Processor:  
- Replace A with @, U with !  
- Reverse, slice, and report length

Part 1B – Lab Mouse Weight Stats:  
- Calculate total, mean, variance, std dev from 5 floats

Part 1C – Cable Segment Classifier:  
- Validate three inputs, classify based on geometric conditions

Part 1D – Oxygen Sensor Data Classifier:  
- Classify list values into four categories  
- Report sorted results and error indices

# Week 4 – Functions and Dictionaries

Learning Objectives:  
- Write reusable functions with parameters and return values  
- Store and retrieve values using dictionaries  
- Apply function composition and iteration

Formative Tasks:  
- Translate codons using a lookup dictionary  
- Implement `translate\_dna()` to convert sequence to protein  
- Fix and explain broken functions

YouTube Video:  
- “Using Functions and Dictionaries for DNA Translation”

# Week 5 – File Handling and Pattern Matching

Learning Objectives:  
- Read and write from text files  
- Use regular expressions to match biological motifs  
- Write formatted summaries from dictionaries

Formative Tasks:  
- Read sequences from a text file  
- Match motifs in sequences using `re.search`  
- Write FASTA or summary outputs

YouTube Video:  
- “Regex and File I/O for Bioinformatics”

# Assignment 2 – Mitochondrial Genome Investigation

Submission: `.py` files + output `.txt` files  
Auto-Marked: Yes (print and file outputs)

Part 2A – Gene Sequence Extraction:  
- Read genome from FASTA  
- Extract gene coordinates from CSV  
- Handle reverse strands  
- Store in dictionary

Part 2B – Translation:  
- Translate gene sequences using codon table  
- Report translation previews

Part 2C – Motif Detection:  
- Search protein sequences for mitochondrial membrane anchor motif using regex

Part 2D – Summary Report:  
- Generate formatted report showing gene ID, protein length, stop codons, motif presence

# Strategy to Reduce Grade Inflation Due to GPT

Students will be explicitly taught how to use generative AI responsibly, including:  
- How to verify GPT output for correctness  
- Common hallucinations and subtle bugs introduced by LLMs  
- Why understanding is essential before using AI-generated code

Assessments are designed to reduce the impact of GPT misuse by:  
- Requiring students to fix broken or incomplete code  
- Using contextually novel problems not directly mirrored in weekly worksheets  
- Requiring code explanation and transformation, not just generation  
- Incorporating randomised data and tightly specified outputs

# Strategy for Auto-Marking Submissions

All assessments are structured to allow for auto-marking using:  
- Predefined test inputs (manual or via `input()`)  
- Capturing and parsing `print()` outputs  
- Comparing against expected outputs using test scripts

Assignment 2 includes file output which is validated using automated file readers.

Marking is resilient to formatting variation through the use of regex and substring validation.

This approach significantly reduces staff marking time while providing timely feedback to students.